

SLATEC2 (AAAAAA through D9UPAK)

Table of Contents

Preface	8
Introduction	9
Using SLATEC Documentation	9
Loading SLATEC Under UNICOS	9
Subroutine Descriptions	11
AAAAAA	12
ACOSH	14
AI	15
AIE	16
ALBETA	17
ALGAMS	18
ALI	19
ALNGAM	20
ALNREL	21
ASINH	22
ATANH	23
AVINT	24
BAKVEC	26
BALANC	28
BALBAK	30
BANDR	32
BANDV	34
BESI	37
BESI0	39
BESI0E	40
BESI1	41
BESI1E	42
BESJ	43
BESJ0	45
BESJ1	46
BESK	47
BESK0	49
BESK0E	50
BESK1	51
BESK1E	52
BESKES	53
BESKNU	54
BESKS	56
BESY	57
BESY0	59
BESY1	60
BETA	61
BETAI	62

BFQAD	63
BI	65
BIE	66
BINOM	68
BINT4	69
BINTK	71
BISECT	73
BLKTRI	75
BNDACC	79
BNDSOL	83
BQR	87
BSKIN	89
BSPDOC	91
BSPDR	96
BSPEV	98
BSPPP	100
BSPVD	102
BSPVN	104
BSQAD	106
BVALU	107
BVSUP	109
C0LGMC	116
CACOS	117
CACOSH	118
CAIRY	119
CARG	122
CASIN	123
CASINH	124
CATAN	125
CATAN2	126
CATANH	127
CAXPY	128
CBABK2	129
CBAL	131
CBESH	133
CBESI	136
CBESJ	139
CBESK	142
CBESY	145
CBETA	148
CBIRY	149
CBLKTR	152
CBRT	156
CCBRT	157
CCHDC	158
CCHDD	160

CCHEX	163
CCHUD	166
CCOPY	168
CCOSH	169
CCOT	170
CDCDOT	171
CDOTC	172
CDOTU	173
CDRIV1	174
CDRIV2	179
CDRIV3	185
CEXPRL	198
CFFTB1	199
CFFTF1	201
CFFTI	203
CFFTI1	204
CG	205
CGAMMA	207
CGAMR	208
CGBCO	209
CGBDI	212
CGBFA	213
CGBMV	215
CGBSL	218
CGECO	220
CGEDI	222
CGEEV	224
CGEFA	226
CGEFS	228
CGEIR	230
CGEMM	232
CGEMV	235
CGERC	237
CGERU	239
CGESL	241
CGTSL	243
CH	245
CHBMV	247
CHEMM	250
CHEMV	253
CHER	255
CHER2	257
CHER2K	259
CHERK	262
CHFDV	265
CHFEV	267

CHICO	269
CHIDI	271
CHIEV	273
CHIFA	275
CHISL	277
CHKDER	279
CHPCO	281
CHPDI	283
CHPFA	285
CHPMV	287
CHPR	289
CHPR2	291
CHPSL	293
CHU	295
CINVIT	296
CLBETA	298
CLNGAM	299
CLNREL	300
CLOG10	301
CMGNBN	302
CNBCO	306
CNBDI	309
CNBFA	310
CNBFS	312
CNBIR	315
CNBSL	318
COMBAK	320
COMHES	322
COMLR	324
COMLR2	326
COMQR	328
COMQR2	330
CORTB	332
CORTH	334
COSDG	336
COSQB	337
COSQF	339
COSQI	341
COST	342
COSTI	344
COT	345
CPBCO	346
CPBDI	348
CPBFA	349
CPBSL	351
CPOCO	353

CPODI	355
CPOFA	357
CPOFS	358
CPOIR	360
CPOSL	362
CPPCO	364
CPPDI	366
CPPFA	368
CPPSL	370
CPQR79	372
CPSI	373
CPTSL	374
CPZERO	375
CQRDC	376
CQRSL	378
CROTG	381
CSCAL	382
CSEVL	383
CSICO	384
CSIDI	386
CSIFA	388
CSINH	390
CSISL	391
CSPCO	393
CSPDI	395
CSPFA	397
CSPSL	399
CSROT	401
CSSCAL	402
CSVDC	403
CSWAP	405
CSYMM	406
CSYR2K	409
CSYRK	412
CTAN	414
CTANH	415
CTBMV	416
CTBSV	419
CTPMV	422
CTPSV	424
CTRCO	426
CTRDI	428
CTRMM	430
CTRMV	433
CTRSL	435
CTRSM	437

CTRSV	440
CV	442
D1MACH	444
D9PAK	446
D9UPAK	447
Disclaimer	448
Structural Keyword Index	449
Date and Revisions	454

Preface

- Scope: SLATEC2 contains brief descriptions ("prologues") for the SLATEC (version 4.1) mathematical library subroutines with names from AAAAAA through D9UPAK.
- Availability: The SLATEC library is downloadable through LINMath (URL: <http://www.llnl.gov/LCdocs/nmg1>) and can be run on all LC production computers.
- Consultant: For help contact the LC customer service and support hotline at 925-422-4531 (open e-mail: lc-hotline@llnl.gov, secure e-mail: lc-hotline@pop.scf.cln).
- Printing: The print file for this document can be found at:
- on the OCF: <http://www.llnl.gov/LCdocs/slatec2/slatec2.pdf>
on the SCF: https://lc.llnl.gov/LCdocs/slatec2/slatec2_scf.pdf

Introduction

Using SLATEC Documentation

Over 1600 pages of online documentation describe the 902 user-callable subroutines available in version 4.1 of the SLATEC library. Because of this unwieldy bulk, the documentation is published in five separate, but interrelated, volumes:

- | | |
|----------------|--|
| <u>SLATEC1</u> | provides introductory information on the whole library, explains the subject categories into which the SLATEC routines are grouped, and includes short descriptions of all routines (alphabetical within each subject category). Every category code is also a link (keyword) for retrieving the brief descriptions of the included routines. SLATEC1 provides the only way to compare related routines by the tasks they perform, rather than just by name. |
| <u>SLATEC2</u> | (THIS DOCUMENT) contains the calling sequence and usage details for each of the 225 subroutines from AAAAAA through D9UPAK, arranged alphabetically by name. Every subroutine name is also a link (keyword) for retrieving the corresponding description if you start at the index. |
| <u>SLATEC3</u> | contains the calling sequence and usage details for each of the 225 subroutines from DACOSH through DS2Y, arranged alphabetically by name. Every subroutine name is also a link (keyword) for retrieving the corresponding description if you start at the index. |
| <u>SLATEC4</u> | contains the calling sequence and usage details for each of the 226 subroutines from DSBMV through RD, arranged alphabetically by name. Every subroutine name is also a link (keyword) for retrieving the corresponding description if you start at the index. |
| <u>SLATEC5</u> | contains the calling sequence and usage details for each of the 226 subroutines from REBAK through ZBIRY, arranged alphabetically by name. Every subroutine name is also a link (keyword) for retrieving the corresponding description if you start at the index. |

You can consult any of these documents from any open machine by running your choice of WWW client and selecting the document you want from the descriptive LC collection directory available at . Or you can specifically request the URL

`http://www.llnl.gov/LCdocs/slatecn`

where `slatecn` is any one of `slatec1` through `slatec5`, depending on which volume you want.

Loading SLATEC Under UNICOS

On LC machines, the SLATEC math library file is called LIBSLATEC.A and has the full pathname

SLATEC2 (AAAAAA through D9UPAK) - 9

`/usr/local/lib/libslatec.a`

The routines in LIBSLATEC.A may use externals in LIBSCI for optimization, and that library is on the default search path (loaded automatically) under UNICOS.

Subroutine Descriptions

AAAAAA

```
      SUBROUTINE AAAAAA (VER)
***BEGIN PROLOGUE  AAAAAA
***PURPOSE  SLATEC Common Mathematical Library disclaimer and version.
***LIBRARY   SLATEC
***CATEGORY  Z
***TYPE      ALL (AAAAAA-A)
***KEYWORDS  DISCLAIMER, DOCUMENTATION, VERSION
***AUTHOR   SLATEC Common Mathematical Library Committee
***DESCRIPTION
```

The SLATEC Common Mathematical Library is issued by the following

Air Force Weapons Laboratory, Albuquerque
Lawrence Livermore National Laboratory, Livermore
Los Alamos National Laboratory, Los Alamos
National Institute of Standards and Technology, Washington
National Energy Research Supercomputer Center, Livermore
Oak Ridge National Laboratory, Oak Ridge
Sandia National Laboratories, Albuquerque
Sandia National Laboratories, Livermore

All questions concerning the distribution of the library should be directed to the NATIONAL ENERGY SOFTWARE CENTER, 9700 Cass Ave., Argonne, Illinois 60439, and not to the authors of the subprograms.

* * * * * Notice * * * * *

This material was prepared as an account of work sponsored by the United States Government. Neither the United States, nor the Department of Energy, nor the Department of Defense, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe upon privately owned rights.

*Usage:

```
      CHARACTER * 16 VER
```

```
      CALL AAAAAA (VER)
```

*Arguments:

VER:OUT will contain the version number of the SLATEC CML.

*Description:

This routine contains the SLATEC Common Mathematical Library disclaimer and can be used to return the library version number.

```
***REFERENCES  Kirby W. Fong, Thomas H. Jefferson, Tokihiko Suyehiro
                and Lee Walton, Guide to the SLATEC Common Mathema-
                tical Library, April 10, 1990.
***ROUTINES CALLED  (NONE)
```

***REVISION HISTORY (YYMMDD)
800424 DATE WRITTEN
890414 REVISION DATE from Version 3.2
890713 Routine modified to return version number. (WRB)
900330 Prologue converted to Version 4.0 format. (BAB)
920501 Reformatted the REFERENCES section. (WRB)
921215 Updated for Version 4.0. (WRB)
930701 Updated for Version 4.1. (WRB)
END PROLOGUE

ACOSH

```
      FUNCTION ACOSH (X)
***BEGIN PROLOGUE  ACOSH
***PURPOSE  Compute the arc hyperbolic cosine.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C4C
***TYPE      SINGLE PRECISION (ACOSH-S, DACOSH-D, CACOSH-C)
***KEYWORDS  ACOSH, ARC HYPERBOLIC COSINE, ELEMENTARY FUNCTIONS, FNLIB,
              INVERSE HYPERBOLIC COSINE
***AUTHOR   Fullerton, W., (LANL)
***DESCRIPTION
```

ACOSH(X) computes the arc hyperbolic cosine of X.

```
***REFERENCES  (NONE)
***ROUTINES CALLED  R1MACH, XERMSG
***REVISION HISTORY  (YYMMDD)
    770401  DATE WRITTEN
    890531  Changed all specific intrinsics to generic.  (WRB)
    890531  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
    900326  Removed duplicate information from DESCRIPTION section.
              (WRB)
END PROLOGUE
```

AI

```
      FUNCTION AI (X)
***BEGIN PROLOGUE  AI
***PURPOSE  Evaluate the Airy function.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C10D
***TYPE      SINGLE PRECISION (AI-S, DAI-D)
***KEYWORDS  AIRY FUNCTION, FNLIB, SPECIAL FUNCTIONS
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION

AI(X) computes the Airy function Ai(X)
Series for AIF          on the interval -1.00000D+00 to  1.00000D+00
                        with weighted error  1.09E-19
                        log weighted error  18.96
                        significant figures required  17.76
                        decimal places required  19.44

Series for AIG          on the interval -1.00000D+00 to  1.00000D+00
                        with weighted error  1.51E-17
                        log weighted error  16.82
                        significant figures required  15.19
                        decimal places required  17.27

***REFERENCES  (NONE)
***ROUTINES CALLED  AIE, CSEVL, INITS, R1MACH, R9AIMP, XERMSG
***REVISION HISTORY  (YYMMDD)
    770701  DATE WRITTEN
    890531  Changed all specific intrinsics to generic.  (WRB)
    890531  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
    900326  Removed duplicate information from DESCRIPTION section.
            (WRB)
    920618  Removed space from variable names.  (RWC, WRB)
END PROLOGUE
```

AIE

```
      FUNCTION AIE (X)
***BEGIN PROLOGUE  AIE
***PURPOSE  Calculate the Airy function for a negative argument and an
              exponentially scaled Airy function for a non-negative
              argument.
***LIBRARY    SLATEC (FNLIB)
***CATEGORY   C10D
***TYPE       SINGLE PRECISION (AIE-S, DAIE-D)
***KEYWORDS   EXPONENTIALLY SCALED AIRY FUNCTION, FNLIB,
              SPECIAL FUNCTIONS
***AUTHOR    Fullerton, W., (LANL)
***DESCRIPTION
```

AIE(X) computes the exponentially scaled Airy function for non-negative X. It evaluates AI(X) for X .LE. 0.0 and EXP(ZETA)*AI(X) for X .GE. 0.0 where ZETA = (2.0/3.0)*(X**1.5).

Series for AIF	on the interval -1.00000D+00 to 1.00000D+00
	with weighted error 1.09E-19
	log weighted error 18.96
	significant figures required 17.76
	decimal places required 19.44

Series for AIG	on the interval -1.00000D+00 to 1.00000D+00
	with weighted error 1.51E-17
	log weighted error 16.82
	significant figures required 15.19
	decimal places required 17.27

Series for AIP	on the interval 0. to 1.00000D+00
	with weighted error 5.10E-17
	log weighted error 16.29
	significant figures required 14.41
	decimal places required 17.06

```
***REFERENCES  (NONE)
***ROUTINES CALLED  CSEVL, INITS, R1MACH, R9AIMP
***REVISION HISTORY  (YYMMDD)
    770701  DATE WRITTEN
    890206  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    920618  Removed space from variable names.  (RWC, WRB)
    END PROLOGUE
```

ALBETA

```
      FUNCTION ALBETA (A, B)
***BEGIN PROLOGUE  ALBETA
***PURPOSE  Compute the natural logarithm of the complete Beta
            function.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C7B
***TYPE      SINGLE PRECISION (ALBETA-S, DLBETA-D, CLBETA-C)
***KEYWORDS  FNLIB, LOGARITHM OF THE COMPLETE BETA FUNCTION,
            SPECIAL FUNCTIONS
***AUTHOR   Fullerton, W., (LANL)
***DESCRIPTION

ALBETA computes the natural log of the complete beta function.

Input Parameters:
      A   real and positive
      B   real and positive

***REFERENCES  (NONE)
***ROUTINES CALLED  ALNGAM, ALNREL, GAMMA, R9LGMC, XERMSG
***REVISION HISTORY  (YYMMDD)
      770701  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890531  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
      900326  Removed duplicate information from DESCRIPTION section.
            (WRB)
      900727  Added EXTERNAL statement.  (WRB)
END PROLOGUE
```

ALGAMS

```
      SUBROUTINE ALGAMS (X, ALGAM, SGNGAM)
***BEGIN PROLOGUE  ALGAMS
***PURPOSE  Compute the logarithm of the absolute value of the Gamma
            function.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C7A
***TYPE      SINGLE PRECISION (ALGAMS-S, DLGAMS-D)
***KEYWORDS  ABSOLUTE VALUE OF THE LOGARITHM OF THE GAMMA FUNCTION,
            FNLIB, SPECIAL FUNCTIONS
***AUTHOR   Fullerton, W., (LANL)
***DESCRIPTION
```

Evaluates the logarithm of the absolute value of the gamma function.

X	- input argument
ALGAM	- result
SGNGAM	- is set to the sign of GAMMA(X) and will be returned at +1.0 or -1.0.

```
***REFERENCES  (NONE)
***ROUTINES CALLED  ALNGAM
***REVISION HISTORY  (YYMMDD)
    770701  DATE WRITTEN
    890531  Changed all specific intrinsics to generic.  (WRB)
    890531  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    END PROLOGUE
```

ALI

```
      FUNCTION ALI (X)
***BEGIN PROLOGUE  ALI
***PURPOSE  Compute the logarithmic integral.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C5
***TYPE      SINGLE PRECISION (ALI-S, DLI-D)
***KEYWORDS  FNLIB, LOGARITHMIC INTEGRAL, SPECIAL FUNCTIONS
***AUTHOR   Fullerton, W., (LANL)
***DESCRIPTION

      ALI(X) computes the logarithmic integral; i.e., the
      integral from 0.0 to X of (1.0/ln(t))dt.

***REFERENCES  (NONE)
***ROUTINES CALLED  EI, XERMSG
***REVISION HISTORY  (YYMMDD)
      770601  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890531  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
      900326  Removed duplicate information from DESCRIPTION section.
              (WRB)
      END PROLOGUE
```

ALNGAM

```
      FUNCTION ALNGAM (X)
***BEGIN PROLOGUE  ALNGAM
***PURPOSE  Compute the logarithm of the absolute value of the Gamma
            function.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C7A
***TYPE      SINGLE PRECISION (ALNGAM-S, DLNGAM-D, CLNGAM-C)
***KEYWORDS  ABSOLUTE VALUE, COMPLETE GAMMA FUNCTION, FNLIB, LOGARITHM,
            SPECIAL FUNCTIONS
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION
```

ALNGAM(X) computes the logarithm of the absolute value of the gamma function at X.

```
***REFERENCES  (NONE)
***ROUTINES CALLED  GAMMA, R1MACH, R9LGMC, XERMSG
***REVISION HISTORY  (YMMDD)
    770601  DATE WRITTEN
    890531  Changed all specific intrinsics to generic.  (WRB)
    890531  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
    900326  Removed duplicate information from DESCRIPTION section.
            (WRB)
    900727  Added EXTERNAL statement.  (WRB)
END PROLOGUE
```

ALNREL

```
      FUNCTION ALNREL (X)
***BEGIN PROLOGUE  ALNREL
***PURPOSE  Evaluate  $\ln(1+X)$  accurate in the sense of relative error.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C4B
***TYPE      SINGLE PRECISION (ALNREL-S, DLNREL-D, CLNREL-C)
***KEYWORDS  ELEMENTARY FUNCTIONS, FNLIB, LOGARITHM
***AUTHOR    Fullerton, W., (LANL)
***DESCRIPTION
```

ALNREL(X) evaluates $\ln(1+X)$ accurately in the sense of relative error when X is very small. This routine must be used to maintain relative error accuracy whenever X is small and accurately known.

```
Series for ALNR           on the interval -3.75000D-01 to  3.75000D-01
                           with weighted error    1.93E-17
                           log weighted error    16.72
                           significant figures required 16.44
                           decimal places required 17.40
```

```
***REFERENCES  (NONE)
***ROUTINES CALLED  CSEVL, INITS, R1MACH, XERMSG
***REVISION HISTORY  (YYMMDD)
    770401  DATE WRITTEN
    890531  Changed all specific intrinsics to generic.  (WRB)
    890531  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
    900326  Removed duplicate information from DESCRIPTION section.
           (WRB)
    END PROLOGUE
```

ASINH

```
      FUNCTION ASINH (X)
***BEGIN PROLOGUE  ASINH
***PURPOSE  Compute the arc hyperbolic sine.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C4C
***TYPE      SINGLE PRECISION (ASINH-S, DASINH-D, CASINH-C)
***KEYWORDS  ARC HYPERBOLIC SINE, ASINH, ELEMENTARY FUNCTIONS, FNLIB,
              INVERSE HYPERBOLIC SINE
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION
```

ASINH(X) computes the arc hyperbolic sine of X.

Series for ASNH	on the interval	0.	to	1.00000D+00
			with weighted error	2.19E-17
			log weighted error	16.66
			significant figures required	15.60
			decimal places required	17.31

```
***REFERENCES  (NONE)
***ROUTINES CALLED  CSEVL, INITS, R1MACH
***REVISION HISTORY  (YYMMDD)
      770401  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890531  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      END PROLOGUE
```

ATANH

```
FUNCTION ATANH (X)
***BEGIN PROLOGUE  ATANH
***PURPOSE  Compute the arc hyperbolic tangent.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C4C
***TYPE      SINGLE PRECISION (ATANH-S, DATANH-D, CATANH-C)
***KEYWORDS  ARC HYPERBOLIC TANGENT, ATANH, ELEMENTARY FUNCTIONS,
              FNLIB, INVERSE HYPERBOLIC TANGENT
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION

    ATANH(X) computes the arc hyperbolic tangent of X.

    Series for ATNH           on the interval  0.           to  2.50000D-01
                                with weighted error  6.70E-18
                                log weighted error  17.17
                                significant figures required  16.01
                                decimal places required  17.76

***REFERENCES  (NONE)
***ROUTINES CALLED  CSEVL, INITS, R1MACH, XERMSG
***REVISION HISTORY  (YYMMDD)
    770401  DATE WRITTEN
    890531  Changed all specific intrinsics to generic.  (WRB)
    890531  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
    900326  Removed duplicate information from DESCRIPTION section.
            (WRB)
END PROLOGUE
```

AVINT

```
SUBROUTINE AVINT (X, Y, N, XLO, XUP, ANS, IERR)
***BEGIN PROLOGUE  AVINT
***PURPOSE  Integrate a function tabulated at arbitrarily spaced
             abscissas using overlapping parabolas.
***LIBRARY   SLATEC
***CATEGORY  H2A1B2
***TYPE      SINGLE PRECISION (AVINT-S, DAVINT-D)
***KEYWORDS  INTEGRATION, QUADRATURE, TABULATED DATA
***AUTHOR   Jones, R. E., (SNLA)
***DESCRIPTION
```

Abstract

AVINT integrates a function tabulated at arbitrarily spaced abscissas. The limits of integration need not coincide with the tabulated abscissas.

A method of overlapping parabolas fitted to the data is used provided that there are at least 3 abscissas between the limits of integration. AVINT also handles two special cases. If the limits of integration are equal, AVINT returns a result of zero regardless of the number of tabulated values. If there are only two function values, AVINT uses the trapezoid rule.

Description of Parameters

The user must dimension all arrays appearing in the call list X(N), Y(N).

Input--

X - real array of abscissas, which must be in increasing order.
Y - real array of functional values. i.e., Y(I)=FUNC(X(I)).
N - the integer number of function values supplied.
N .GE. 2 unless XLO = XUP.
XLO - real lower limit of integration.
XUP - real upper limit of integration.
Must have XLO .LE. XUP.

Output--

ANS - computed approximate value of integral
IERR - a status code
--normal code
=1 means the requested integration was performed.
--abnormal codes
=2 means XUP was less than XLO.
=3 means the number of X(I) between XLO and XUP (inclusive) was less than 3 and neither of the two special cases described in the Abstract occurred.
No integration was performed.
=4 means the restriction X(I+1) .GT. X(I) was violated.
=5 means the number N of function values was .LT. 2.
ANS is set to zero if IERR=2,3,4, or 5.

AVINT is documented completely in SC-M-69-335
Original program from "Numerical Integration" by Davis & Rabinowitz.

Adaptation and modifications for Sandia Mathematical Program
Library by Rondall E. Jones.

```
***REFERENCES  R. E. Jones, Approximate integrator of functions
                tabulated at arbitrarily spaced abscissas,
                Report SC-M-69-335, Sandia Laboratories, 1969.
***ROUTINES CALLED  XERMSG
***REVISION HISTORY  (YYMMDD)
  690901  DATE WRITTEN
  890831  Modified array declarations.  (WRB)
  890831  REVISION DATE from Version 3.2
  891214  Prologue converted to Version 4.0 format.  (BAB)
  900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
  900326  Removed duplicate information from DESCRIPTION section.
          (WRB)
  920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

BAKVEC

```
SUBROUTINE BAKVEC (NM, N, T, E, M, Z, IERR)
***BEGIN PROLOGUE  BAKVEC
***PURPOSE  Form the eigenvectors of a certain real non-symmetric
             tridiagonal matrix from a symmetric tridiagonal matrix
             output from FIGI.
***LIBRARY   SLATEC (EISPACK)
***CATEGORY  D4C4
***TYPE      SINGLE PRECISION (BAKVEC-S)
***KEYWORDS  EIGENVECTORS, EISPACK
***AUTHOR    Smith, B. T., et al.
***DESCRIPTION
```

This subroutine forms the eigenvectors of a NONSYMMETRIC TRIDIAGONAL matrix by back transforming those of the corresponding symmetric matrix determined by FIGI.

On INPUT

NM must be set to the row dimension of the two-dimensional array parameters, T and Z, as declared in the calling program dimension statement. NM is an INTEGER variable.

N is the order of the matrix T. N is an INTEGER variable. N must be less than or equal to NM.

T contains the nonsymmetric matrix. Its subdiagonal is stored in the last N-1 positions of the first column, its diagonal in the N positions of the second column, and its superdiagonal in the first N-1 positions of the third column. T(1,1) and T(N,3) are arbitrary. T is a two-dimensional REAL array, dimensioned T(NM,3).

E contains the subdiagonal elements of the symmetric matrix in its last N-1 positions. E(1) is arbitrary. E is a one-dimensional REAL array, dimensioned E(N).

M is the number of eigenvectors to be back transformed. M is an INTEGER variable.

Z contains the eigenvectors to be back transformed in its first M columns. Z is a two-dimensional REAL array, dimensioned Z(NM,M).

On OUTPUT

T is unaltered.

E is destroyed.

Z contains the transformed eigenvectors in its first M columns.

IERR is an INTEGER flag set to
Zero for normal return,
2*N+I if E(I) is zero with T(I,1) or T(I-1,3) non-zero.
In this case, the symmetric matrix is not similar
to the original matrix, and the eigenvectors

cannot be found by this program.

Questions and comments should be directed to B. S. Garbow,
APPLIED MATHEMATICS DIVISION, ARGONNE NATIONAL LABORATORY

***REFERENCES B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow,
Y. Ikebe, V. C. Klema and C. B. Moler, Matrix Eigen-
system Routines - EISPACK Guide, Springer-Verlag,
1976.

***ROUTINES CALLED (NONE)

***REVISION HISTORY (YYMMDD)

760101 DATE WRITTEN

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

BALANC

```
SUBROUTINE BALANC (NM, N, A, LOW, IGH, SCALE)
***BEGIN PROLOGUE  BALANC
***PURPOSE  Balance a real general matrix and isolate eigenvalues
            whenever possible.
***LIBRARY   SLATEC (EISPACK)
***CATEGORY  D4C1A
***TYPE      SINGLE PRECISION (BALANC-S, CBAL-C)
***KEYWORDS  EIGENVECTORS, EISPACK
***AUTHOR    Smith, B. T., et al.
***DESCRIPTION
```

This subroutine is a translation of the ALGOL procedure BALANCE, NUM. MATH. 13, 293-304(1969) by Parlett and Reinsch. HANDBOOK FOR AUTO. COMP., Vol.II-LINEAR ALGEBRA, 315-326(1971).

This subroutine balances a REAL matrix and isolates eigenvalues whenever possible.

On INPUT

NM must be set to the row dimension of the two-dimensional array parameter, A, as declared in the calling program dimension statement. NM is an INTEGER variable.

N is the order of the matrix A. N is an INTEGER variable. N must be less than or equal to NM.

A contains the input matrix to be balanced. A is a two-dimensional REAL array, dimensioned A(NM,N).

On OUTPUT

A contains the balanced matrix.

LOW and IGH are two INTEGER variables such that A(I,J) is equal to zero if

- (1) I is greater than J and
- (2) J=1,...,LOW-1 or I=IGH+1,...,N.

SCALE contains information determining the permutations and scaling factors used. SCALE is a one-dimensional REAL array, dimensioned SCALE(N).

Suppose that the principal submatrix in rows LOW through IGH has been balanced, that P(J) denotes the index interchanged with J during the permutation step, and that the elements of the diagonal matrix used are denoted by D(I,J). Then

$$\begin{aligned} \text{SCALE}(J) &= P(J), & \text{for } J &= 1, \dots, \text{LOW}-1 \\ &= D(J,J), & J &= \text{LOW}, \dots, \text{IGH} \\ &= P(J) & J &= \text{IGH}+1, \dots, N. \end{aligned}$$

The order in which the interchanges are made is N to IGH+1, then 1 TO LOW-1.

Note that 1 is returned for IGH if IGH is zero formally.

The ALGOL procedure EXC contained in BALANCE appears in

BALANC in line. (Note that the ALGOL roles of identifiers K,L have been reversed.)

Questions and comments should be directed to B. S. Garbow,
Applied Mathematics Division, ARGONNE NATIONAL LABORATORY

***REFERENCES B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow,
Y. Ikebe, V. C. Klema and C. B. Moler, Matrix Eigen-
system Routines - EISPACK Guide, Springer-Verlag,
1976.

***ROUTINES CALLED (NONE)

***REVISION HISTORY (YYMMDD)

760101 DATE WRITTEN

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

BALBAK

```
SUBROUTINE BALBAK (NM, N, LOW, IGH, SCALE, M, Z)
***BEGIN PROLOGUE  BALBAK
***PURPOSE  Form the eigenvectors of a real general matrix from the
            eigenvectors of matrix output from BALANC.
***LIBRARY   SLATEC (EISPACK)
***CATEGORY  D4C4
***TYPE      SINGLE PRECISION (BALBAK-S, CBABK2-C)
***KEYWORDS  EIGENVECTORS, EISPACK
***AUTHOR    Smith, B. T., et al.
***DESCRIPTION
```

This subroutine is a translation of the ALGOL procedure BALBAK, NUM. MATH. 13, 293-304(1969) by Parlett and Reinsch. HANDBOOK FOR AUTO. COMP., Vol.II-LINEAR ALGEBRA, 315-326(1971).

This subroutine forms the eigenvectors of a REAL GENERAL matrix by back transforming those of the corresponding balanced matrix determined by BALANC.

On INPUT

NM must be set to the row dimension of the two-dimensional array parameter, Z, as declared in the calling program dimension statement. NM is an INTEGER variable.

N is the number of components of the vectors in matrix Z. N is an INTEGER variable. N must be less than or equal to NM.

LOW and IGH are INTEGER variables determined by BALANC.

SCALE contains information determining the permutations and scaling factors used by BALANC. SCALE is a one-dimensional REAL array, dimensioned SCALE(N).

M is the number of columns of Z to be back transformed. M is an INTEGER variable.

Z contains the real and imaginary parts of the eigenvectors to be back transformed in its first M columns. Z is a two-dimensional REAL array, dimensioned Z(NM,M).

On OUTPUT

Z contains the real and imaginary parts of the transformed eigenvectors in its first M columns.

Questions and comments should be directed to B. S. Garbow, Applied Mathematics Division, ARGONNE NATIONAL LABORATORY

```
-----
***REFERENCES  B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow,
                Y. Ikebe, V. C. Klema and C. B. Moler, Matrix Eigen-
                system Routines - EISPACK Guide, Springer-Verlag,
                1976.
***ROUTINES CALLED  (NONE)
```

```
***REVISION HISTORY  (YYMMDD)
 760101  DATE WRITTEN
 890831  Modified array declarations.  (WRB)
 890831  REVISION DATE from Version 3.2
 891214  Prologue converted to Version 4.0 format.  (BAB)
 920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

BANDR

```
SUBROUTINE BANDR (NM, N, MB, A, D, E, E2, MATZ, Z)
***BEGIN PROLOGUE  BANDR
***PURPOSE  Reduce a real symmetric band matrix to symmetric
             tridiagonal matrix and, optionally, accumulate
             orthogonal similarity transformations.
***LIBRARY   SLATEC (EISPACK)
***CATEGORY  D4C1B1
***TYPE      SINGLE PRECISION (BANDR-S)
***KEYWORDS  EIGENVALUES, EIGENVECTORS, EISPACK
***AUTHOR    Smith, B. T., et al.
***DESCRIPTION
```

This subroutine is a translation of the ALGOL procedure BANDRD, NUM. MATH. 12, 231-241(1968) by Schwarz. HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 273-283(1971).

This subroutine reduces a REAL SYMMETRIC BAND matrix to a symmetric tridiagonal matrix using and optionally accumulating orthogonal similarity transformations.

On INPUT

NM must be set to the row dimension of the two-dimensional array parameters, A and Z, as declared in the calling program dimension statement. NM is an INTEGER variable.

N is the order of the matrix A. N is an INTEGER variable. N must be less than or equal to NM.

MB is the (half) band width of the matrix, defined as the number of adjacent diagonals, including the principal diagonal, required to specify the non-zero portion of the lower triangle of the matrix. MB is less than or equal to N. MB is an INTEGER variable.

A contains the lower triangle of the real symmetric band matrix. Its lowest subdiagonal is stored in the last N+1-MB positions of the first column, its next subdiagonal in the last N+2-MB positions of the second column, further subdiagonals similarly, and finally its principal diagonal in the N positions of the last column. Contents of storage locations not part of the matrix are arbitrary. A is a two-dimensional REAL array, dimensioned A(NM,MB).

MATZ should be set to .TRUE. if the transformation matrix is to be accumulated, and to .FALSE. otherwise. MATZ is a LOGICAL variable.

On OUTPUT

A has been destroyed, except for its last two columns which contain a copy of the tridiagonal matrix.

D contains the diagonal elements of the tridiagonal matrix. D is a one-dimensional REAL array, dimensioned D(N).

E contains the subdiagonal elements of the tridiagonal matrix in its last N-1 positions. E(1) is set to zero. E is a one-dimensional REAL array, dimensioned E(N).

E2 contains the squares of the corresponding elements of E. E2 may coincide with E if the squares are not needed. E2 is a one-dimensional REAL array, dimensioned E2(N).

Z contains the orthogonal transformation matrix produced in the reduction if MATZ has been set to .TRUE. Otherwise, Z is not referenced. Z is a two-dimensional REAL array, dimensioned Z(NM,N).

Questions and comments should be directed to B. S. Garbow,
Applied Mathematics Division, ARGONNE NATIONAL LABORATORY

***REFERENCES B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow,
Y. Ikebe, V. C. Klema and C. B. Moler, Matrix Eigen-
system Routines - EISPACK Guide, Springer-Verlag,
1976.

***ROUTINES CALLED (NONE)

***REVISION HISTORY (YYMMDD)

760101 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

BANDV

```
SUBROUTINE BANDV (NM, N, MBW, A, E21, M, W, Z, IERR, NV, RV, RV6)
***BEGIN PROLOGUE  BANDV
***PURPOSE  Form the eigenvectors of a real symmetric band matrix
              associated with a set of ordered approximate eigenvalues
              by inverse iteration.
***LIBRARY   SLATEC (EISPACK)
***CATEGORY  D4C3
***TYPE      SINGLE PRECISION (BANDV-S)
***KEYWORDS  EIGENVECTORS, EISPACK
***AUTHOR    Smith, B. T., et al.
***DESCRIPTION
```

This subroutine finds those eigenvectors of a REAL SYMMETRIC BAND matrix corresponding to specified eigenvalues, using inverse iteration. The subroutine may also be used to solve systems of linear equations with a symmetric or non-symmetric band coefficient matrix.

On INPUT

NM must be set to the row dimension of the two-dimensional array parameters, A and Z, as declared in the calling program dimension statement. NM is an INTEGER variable.

N is the order of the matrix A. N is an INTEGER variable. N must be less than or equal to NM.

MBW is the number of columns of the array A used to store the band matrix. If the matrix is symmetric, MBW is its (half) band width, denoted MB and defined as the number of adjacent diagonals, including the principal diagonal, required to specify the non-zero portion of the lower triangle of the matrix. If the subroutine is being used to solve systems of linear equations and the coefficient matrix is not symmetric, it must however have the same number of adjacent diagonals above the main diagonal as below, and in this case, $MBW=2*MB-1$. MBW is an INTEGER variable. MB must not be greater than N.

A contains the lower triangle of the symmetric band input matrix stored as an N by MB array. Its lowest subdiagonal is stored in the last $N+1-MB$ positions of the first column, its next subdiagonal in the last $N+2-MB$ positions of the second column, further subdiagonals similarly, and finally its principal diagonal in the N positions of column MB. If the subroutine is being used to solve systems of linear equations and the coefficient matrix is not symmetric, A is N by $2*MB-1$ instead with lower triangle as above and with its first superdiagonal stored in the first $N-1$ positions of column $MB+1$, its second superdiagonal in the first $N-2$ positions of column $MB+2$, further superdiagonals similarly, and finally its highest superdiagonal in the first $N+1-MB$ positions of the last column. Contents of storage locations not part of the matrix are arbitrary. A is a two-dimensional REAL array, dimensioned A(NM,MBW).

E21 specifies the ordering of the eigenvalues and contains
0.0E0 if the eigenvalues are in ascending order, or
2.0E0 if the eigenvalues are in descending order.
If the subroutine is being used to solve systems of linear
equations, E21 should be set to 1.0E0 if the coefficient
matrix is symmetric and to -1.0E0 if not. E21 is a REAL
variable.

M is the number of specified eigenvalues or the number of
systems of linear equations. M is an INTEGER variable.

W contains the M eigenvalues in ascending or descending order.
If the subroutine is being used to solve systems of linear
equations $(A-W(J)*I)*X(J)=B(J)$, where I is the identity
matrix, W(J) should be set accordingly, for $J=1,2,\dots,M$.
W is a one-dimensional REAL array, dimensioned W(M).

Z contains the constant matrix columns $(B(J), J=1,2,\dots,M)$, if
the subroutine is used to solve systems of linear equations.
Z is a two-dimensional REAL array, dimensioned Z(NM,M).

NV must be set to the dimension of the array parameter RV
as declared in the calling program dimension statement.
NV is an INTEGER variable.

On OUTPUT

A and W are unaltered.

Z contains the associated set of orthogonal eigenvectors.
Any vector which fails to converge is set to zero. If the
subroutine is used to solve systems of linear equations,
Z contains the solution matrix columns $(X(J), J=1,2,\dots,M)$.

IERR is an INTEGER flag set to
Zero for normal return,
-J if the eigenvector corresponding to the J-th
eigenvalue fails to converge, or if the J-th
system of linear equations is nearly singular.

RV and RV6 are temporary storage arrays. If the subroutine
is being used to solve systems of linear equations, the
determinant (up to sign) of $A-W(M)*I$ is available, upon
return, as the product of the first N elements of RV.
RV and RV6 are one-dimensional REAL arrays. Note that RV
is dimensioned RV(NV), where NV must be at least $N*(2*MB-1)$.
RV6 is dimensioned RV6(N).

Questions and comments should be directed to B. S. Garbow,
Applied Mathematics Division, ARGONNE NATIONAL LABORATORY

***REFERENCES B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow,
Y. Ikebe, V. C. Klema and C. B. Moler, Matrix Eigen-
system Routines - EISPACK Guide, Springer-Verlag,
1976.

***ROUTINES CALLED (NONE)

***REVISION HISTORY (YYMMDD)

760101 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

SLATEC2 (AAAAAA through D9UPAK) - 35

890831 Modified array declarations. (WRB)
890831 REVISION DATE from Version 3.2
891214 Prologue converted to Version 4.0 format. (BAB)
920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

BESI

```
SUBROUTINE BESI (X, ALPHA, KODE, N, Y, NZ)
***BEGIN PROLOGUE  BESI
***PURPOSE  Compute an N member sequence of I Bessel functions
             I/SUB(ALPHA+K-1)/(X), K=1,...,N or scaled Bessel functions
             EXP(-X)*I/SUB(ALPHA+K-1)/(X), K=1,...,N for non-negative
             ALPHA and X.
***LIBRARY    SLATEC
***CATEGORY   C10B3
***TYPE       SINGLE PRECISION (BESI-S, DBESI-D)
***KEYWORDS   I BESSEL FUNCTION, SPECIAL FUNCTIONS
***AUTHOR     Amos, D. E., (SNLA)
               Daniel, S. L., (SNLA)
***DESCRIPTION
```

Abstract

BESI computes an N member sequence of I Bessel functions $I/\text{sub}(\text{ALPHA}+K-1)/(X)$, $K=1,\dots,N$ or scaled Bessel functions $\text{EXP}(-X)*I/\text{sub}(\text{ALPHA}+K-1)/(X)$, $K=1,\dots,N$ for non-negative ALPHA and X. A combination of the power series, the asymptotic expansion for X to infinity, and the uniform asymptotic expansion for NU to infinity are applied over subdivisions of the (NU,X) plane. For values not covered by one of these formulae, the order is incremented by an integer so that one of these formulae apply. Backward recursion is used to reduce orders by integer values. The asymptotic expansion for X to infinity is used only when the entire sequence (specifically the last member) lies within the region covered by the expansion. Leading terms of these expansions are used to test for over or underflow where appropriate. If a sequence is requested and the last member would underflow, the result is set to zero and the next lower order tried, etc., until a member comes on scale or all are set to zero. An overflow cannot occur with scaling.

Description of Arguments

Input

X - X .GE. 0.0E0
ALPHA - order of first member of the sequence,
 ALPHA .GE. 0.0E0
KODE - a parameter to indicate the scaling option
 KODE=1 returns
 $Y(K) = I/\text{sub}(\text{ALPHA}+K-1)/(X),$
 $K=1,\dots,N$
 KODE=2 returns
 $Y(K) = \text{EXP}(-X)*I/\text{sub}(\text{ALPHA}+K-1)/(X),$
 $K=1,\dots,N$
N - number of members in the sequence, N .GE. 1

Output

Y - a vector whose first N components contain
 values for $I/\text{sub}(\text{ALPHA}+K-1)/(X)$ or scaled
 values for $\text{EXP}(-X)*I/\text{sub}(\text{ALPHA}+K-1)/(X),$
 $K=1,\dots,N$ depending on KODE
NZ - number of components of Y set to zero due to
 underflow,

NZ=0 , normal return, computation completed
NZ .NE. 0, last NZ components of Y set to zero,
Y(K)=0.0E0, K=N-NZ+1,...,N.

Error Conditions

Improper input arguments - a fatal error
Overflow with KODE=1 - a fatal error
Underflow - a non-fatal error (NZ .NE. 0)

***REFERENCES D. E. Amos, S. L. Daniel and M. K. Weston, CDC 6600
subroutines IBESS and JBESS for Bessel functions
I(NU,X) and J(NU,X), X .GE. 0, NU .GE. 0, ACM
Transactions on Mathematical Software 3, (1977),
pp. 76-92.
F. W. J. Olver, Tables of Bessel Functions of Moderate
or Large Orders, NPL Mathematical Tables 6, Her
Majesty's Stationery Office, London, 1962.

***ROUTINES CALLED ALNGAM, ASYIK, ILMACH, RLMACH, XERMSG

***REVISION HISTORY (YYMMDD)

750101 DATE WRITTEN
890531 Changed all specific intrinsics to generic. (WRB)
890531 REVISION DATE from Version 3.2
891214 Prologue converted to Version 4.0 format. (BAB)
900315 CALLs to XERROR changed to CALLs to XERMSG. (THJ)
900326 Removed duplicate information from DESCRIPTION section.
(WRB)
920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

BESIO

```
      FUNCTION BESIO (X)
***BEGIN PROLOGUE  BESIO
***PURPOSE  Compute the hyperbolic Bessel function of the first kind
             of order zero.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C10B1
***TYPE      SINGLE PRECISION (BESIO-S, DBESIO-D)
***KEYWORDS  FIRST KIND, FNLIB, HYPERBOLIC BESSEL FUNCTION,
             MODIFIED BESSEL FUNCTION, ORDER ZERO, SPECIAL FUNCTIONS
***AUTHOR   Fullerton, W., (LANL)
***DESCRIPTION

BESIO(X) computes the modified (hyperbolic) Bessel function
of the first kind of order zero and real argument X.

Series for BIO          on the interval  0.          to  9.00000D+00
                               with weighted error  2.46E-18
                               log weighted error  17.61
                               significant figures required  17.90
                               decimal places required  18.15

***REFERENCES  (NONE)
***ROUTINES CALLED  BESIOE, CSEVL, INITS, R1MACH, XERMSG
***REVISION HISTORY  (YYMMDD)
    770401  DATE WRITTEN
    890531  Changed all specific intrinsics to generic.  (WRB)
    890531  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
    900326  Removed duplicate information from DESCRIPTION section.
            (WRB)
END PROLOGUE
```

BESIOE

```
      FUNCTION BESIOE (X)
***BEGIN PROLOGUE  BESIOE
***PURPOSE  Compute the exponentially scaled modified (hyperbolic)
             Bessel function of the first kind of order zero.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C10B1
***TYPE      SINGLE PRECISION (BESIOE-S, DBSIOE-D)
***KEYWORDS  EXPONENTIALLY SCALED, FIRST KIND, FNLIB,
             HYPERBOLIC BESSEL FUNCTION, MODIFIED BESSEL FUNCTION,
             ORDER ZERO, SPECIAL FUNCTIONS
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION

BESIOE(X) calculates the exponentially scaled modified (hyperbolic)
Bessel function of the first kind of order zero for real argument X;
i.e.,  $\text{EXP}(-\text{ABS}(X)) * \text{I0}(X)$ .
```

```
Series for BIO          on the interval  0.          to  9.00000D+00
                        with weighted error  2.46E-18
                        log weighted error  17.61
                        significant figures required  17.90
                        decimal places required  18.15
```

```
Series for AI0          on the interval  1.25000D-01 to  3.33333D-01
                        with weighted error  7.87E-17
                        log weighted error  16.10
                        significant figures required  14.69
                        decimal places required  16.76
```

```
Series for AI02         on the interval  0.          to  1.25000D-01
                        with weighted error  3.79E-17
                        log weighted error  16.42
                        significant figures required  14.86
                        decimal places required  17.09
```

```
***REFERENCES  (NONE)
***ROUTINES CALLED  CSEVL, INITS, R1MACH
***REVISION HISTORY  (YYMMDD)
    770701  DATE WRITTEN
    890313  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    END PROLOGUE
```

BES11

```
      FUNCTION BES11 (X)
***BEGIN PROLOGUE  BES11
***PURPOSE  Compute the modified (hyperbolic) Bessel function of the
             first kind of order one.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C10B1
***TYPE      SINGLE PRECISION (BES11-S, DBES11-D)
***KEYWORDS  FIRST KIND, FNLIB, HYPERBOLIC BESSEL FUNCTION,
             MODIFIED BESSEL FUNCTION, ORDER ONE, SPECIAL FUNCTIONS
***AUTHOR   Fullerton, W., (LANL)
***DESCRIPTION
```

BES11(X) calculates the modified (hyperbolic) Bessel function
of the first kind of order one for real argument X.

```
Series for B11          on the interval  0.          to  9.00000D+00
                                with weighted error  2.40E-17
                                log weighted error  16.62
                                significant figures required  16.23
                                decimal places required  17.14
```

```
***REFERENCES  (NONE)
***ROUTINES CALLED  BES11E, CSEVL, INITS, R1MACH, XERMSG
***REVISION HISTORY  (YYMMDD)
    770401  DATE WRITTEN
    890531  Changed all specific intrinsics to generic.  (WRB)
    890531  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
    900326  Removed duplicate information from DESCRIPTION section.
            (WRB)
END PROLOGUE
```

BES1E

```
FUNCTION BES1E (X)
***BEGIN PROLOGUE  BES1E
***PURPOSE  Compute the exponentially scaled modified (hyperbolic)
            Bessel function of the first kind of order one.
***LIBRARY  SLATEC (FNLIB)
***CATEGORY  C10B1
***TYPE     SINGLE PRECISION (BES1E-S, DBS1E-D)
***KEYWORDS  EXPONENTIALLY SCALED, FIRST KIND, FNLIB,
            HYPERBOLIC BESSEL FUNCTION, MODIFIED BESSEL FUNCTION,
            ORDER ONE, SPECIAL FUNCTIONS
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION

BES1E(X) calculates the exponentially scaled modified (hyperbolic)
Bessel function of the first kind of order one for real argument X;
i.e.,  $\text{EXP}(-\text{ABS}(X)) * \text{I1}(X)$ .

Series for BI1          on the interval  0.          to 9.00000D+00
                        with weighted error 2.40E-17
                        log weighted error 16.62
                        significant figures required 16.23
                        decimal places required 17.14

Series for AI1          on the interval 1.25000D-01 to 3.33333D-01
                        with weighted error 6.98E-17
                        log weighted error 16.16
                        significant figures required 14.53
                        decimal places required 16.82

Series for AI12         on the interval 0.          to 1.25000D-01
                        with weighted error 3.55E-17
                        log weighted error 16.45
                        significant figures required 14.69
                        decimal places required 17.12

***REFERENCES  (NONE)
***ROUTINES CALLED  CSEVL, INITS, R1MACH, XERMSG
***REVISION HISTORY  (YYMMDD)
    770401  DATE WRITTEN
    890210  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
    900326  Removed duplicate information from DESCRIPTION section.
            (WRB)
    920618  Removed space from variable names.  (RWC, WRB)
END PROLOGUE
```

BESJ

```
SUBROUTINE BESJ (X, ALPHA, N, Y, NZ)
***BEGIN PROLOGUE  BESJ
***PURPOSE  Compute an N member sequence of J Bessel functions
             J/SUB(ALPHA+K-1)/(X), K=1,...,N for non-negative ALPHA
             and X.
***LIBRARY  SLATEC
***CATEGORY  C10A3
***TYPE      SINGLE PRECISION (BESJ-S, DBESJ-D)
***KEYWORDS  J BESSEL FUNCTION, SPECIAL FUNCTIONS
***AUTHOR  Amos, D. E., (SNLA)
            Daniel, S. L., (SNLA)
            Weston, M. K., (SNLA)
***DESCRIPTION
```

Abstract

BESJ computes an N member sequence of J Bessel functions $J/\text{sub}(\text{ALPHA}+K-1)/(X)$, $K=1,\dots,N$ for non-negative ALPHA and X. A combination of the power series, the asymptotic expansion for X to infinity and the uniform asymptotic expansion for NU to infinity are applied over subdivisions of the (NU,X) plane. For values of (NU,X) not covered by one of these formulae, the order is incremented or decremented by integer values into a region where one of the formulae apply. Backward recursion is applied to reduce orders by integer values except where the entire sequence lies in the oscillatory region. In this case forward recursion is stable and values from the asymptotic expansion for X to infinity start the recursion when it is efficient to do so. Leading terms of the series and uniform expansion are tested for underflow. If a sequence is requested and the last member would underflow, the result is set to zero and the next lower order tried, etc., until a member comes on scale or all members are set to zero. Overflow cannot occur.

Description of Arguments

Input

X - X .GE. 0.0E0
ALPHA - order of first member of the sequence,
ALPHA .GE. 0.0E0
N - number of members in the sequence, N .GE. 1

Output

Y - a vector whose first N components contain
values for $J/\text{sub}(\text{ALPHA}+K-1)/(X)$, $K=1,\dots,N$
NZ - number of components of Y set to zero due to
underflow,
NZ=0 , normal return, computation completed
NZ .NE. 0, last NZ components of Y set to zero,
 $Y(K)=0.0E0$, $K=N-NZ+1,\dots,N$.

Error Conditions

Improper input arguments - a fatal error
Underflow - a non-fatal error (NZ .NE. 0)

***REFERENCES D. E. Amos, S. L. Daniel and M. K. Weston, CDC 6600

subroutines IBESS and JBESS for Bessel functions
 $I(\text{NU}, X)$ and $J(\text{NU}, X)$, $X \geq 0$, $\text{NU} \geq 0$, ACM
Transactions on Mathematical Software 3, (1977),
pp. 76-92.

F. W. J. Olver, Tables of Bessel Functions of Moderate
or Large Orders, NPL Mathematical Tables 6, Her
Majesty's Stationery Office, London, 1962.

***ROUTINES CALLED ALNGAM, ASYJY, ILMACH, JAIRY, RLMACH, XERMSG
***REVISION HISTORY (YYMMDD)
750101 DATE WRITTEN
890531 Changed all specific intrinsics to generic. (WRB)
890531 REVISION DATE from Version 3.2
891214 Prologue converted to Version 4.0 format. (BAB)
900315 CALLs to XERROR changed to CALLs to XERMSG. (THJ)
900326 Removed duplicate information from DESCRIPTION section.
(WRB)
920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

BESJ0

```
      FUNCTION BESJ0 (X)
***BEGIN PROLOGUE  BESJ0
***PURPOSE  Compute the Bessel function of the first kind of order
            zero.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C10A1
***TYPE      SINGLE PRECISION (BESJ0-S, DBESJ0-D)
***KEYWORDS  BESSEL FUNCTION, FIRST KIND, FNLIB, ORDER ZERO,
            SPECIAL FUNCTIONS
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION
```

BESJ0(X) calculates the Bessel function of the first kind of order zero for real argument X.

```
Series for BJ0          on the interval  0.          to 1.60000D+01
                        with weighted error 7.47E-18
                        log weighted error 17.13
                        significant figures required 16.98
                        decimal places required 17.68
```

```
Series for BM0          on the interval  0.          to 6.25000D-02
                        with weighted error 4.98E-17
                        log weighted error 16.30
                        significant figures required 14.97
                        decimal places required 16.96
```

```
Series for BTH0         on the interval  0.          to 6.25000D-02
                        with weighted error 3.67E-17
                        log weighted error 16.44
                        significant figures required 15.53
                        decimal places required 17.13
```

```
***REFERENCES  (NONE)
***ROUTINES CALLED  CSEVL, INITS, R1MACH, XERMSG
***REVISION HISTORY  (YYMMDD)
    770401  DATE WRITTEN
    890210  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
    900326  Removed duplicate information from DESCRIPTION section.
            (WRB)
END PROLOGUE
```

BESJ1

```
      FUNCTION BESJ1 (X)
***BEGIN PROLOGUE  BESJ1
***PURPOSE  Compute the Bessel function of the first kind of order one.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C10A1
***TYPE      SINGLE PRECISION (BESJ1-S, DBESJ1-D)
***KEYWORDS  BESSEL FUNCTION, FIRST KIND, FNLIB, ORDER ONE,
              SPECIAL FUNCTIONS
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION
```

BESJ1(X) calculates the Bessel function of the first kind of order one for real argument X.

```
Series for BJ1          on the interval  0.          to 1.60000D+01
                        with weighted error  4.48E-17
                        log weighted error  16.35
                        significant figures required 15.77
                        decimal places required 16.89
```

```
Series for BM1          on the interval  0.          to 6.25000D-02
                        with weighted error  5.61E-17
                        log weighted error  16.25
                        significant figures required 14.97
                        decimal places required 16.91
```

```
Series for BTH1         on the interval  0.          to 6.25000D-02
                        with weighted error  4.10E-17
                        log weighted error  16.39
                        significant figures required 15.96
                        decimal places required 17.08
```

```
***REFERENCES  (NONE)
***ROUTINES CALLED  CSEVL, INITS, R1MACH, XERMSG
***REVISION HISTORY  (YYMMDD)
      780601  DATE WRITTEN
      890210  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
      900326  Removed duplicate information from DESCRIPTION section.
              (WRB)
      END PROLOGUE
```

BESK

```

SUBROUTINE BESK (X, FNU, KODE, N, Y, NZ)
***BEGIN PROLOGUE  BESK
***PURPOSE  Implement forward recursion on the three term recursion
            relation for a sequence of non-negative order Bessel
            functions K/SUB(FNU+I-1)/(X), or scaled Bessel functions
            EXP(X)*K/SUB(FNU+I-1)/(X), I=1,...,N for real, positive
            X and non-negative orders FNU.
***LIBRARY    SLATEC
***CATEGORY   C10B3
***TYPE       SINGLE PRECISION (BESK-S, DBESK-D)
***KEYWORDS   K BESSEL FUNCTION, SPECIAL FUNCTIONS
***AUTHOR     Amos, D. E., (SNLA)
***DESCRIPTION

```

Abstract

BESK implements forward recursion on the three term recursion relation for a sequence of non-negative order Bessel functions $K_{\text{FNU}+I-1}(X)$, or scaled Bessel functions $\text{EXP}(X) * K_{\text{FNU}+I-1}(X)$, $I=1, \dots, N$ for real $X > 0.0E0$ and non-negative orders FNU. If FNU \leq NULIM, orders FNU and FNU+1 are obtained from BESKNU to start the recursion. If FNU \geq NULIM, the uniform asymptotic expansion is used for orders FNU and FNU+1 to start the recursion. NULIM is 35 or 70 depending on whether N=1 or $N \geq 2$. Under and overflow tests are made on the leading term of the asymptotic expansion before any extensive computation is done.

Description of Arguments

Input

```

X      - X .GT. 0.0E0
FNU    - order of the initial K function, FNU .GE. 0.0E0
KODE   - a parameter to indicate the scaling option
        KODE=1 returns  $Y(I) = \frac{K}{\text{sub}(FNU+I-1)/(X)},$ 
           I=1,...,N
        KODE=2 returns  $Y(I) = \frac{\text{EXP}(X)*K}{\text{sub}(FNU+I-1)/(X)},$ 
           I=1,...,N
N      - number of members in the sequence, N .GE. 1

```

Output

```

Y      - a vector whose first n components contain values
        for the sequence
        Y(I)=      K/sub(FNU+I-1)/(X), I=1,...,N   or
        Y(I)=EXP(X)*K/sub(FNU+I-1)/(X), I=1,...,N
        depending on CODE
NZ     - number of components of Y set to zero due to
        underflow with CODE=1,
        NZ=0      , normal return, computation completed
        NZ .NE. 0, first NZ components of Y set to zero
                   due to underflow, Y(I)=0.0E0, I=1,...,NZ

```

Error Conditions

Improper input arguments - a fatal error
 Overflow - a fatal error
 Underflow with `KODE=1` - a non-fatal error (NZ .NE. 0)

```

***REFERENCES  F. W. J. Olver, Tables of Bessel Functions of Moderate
                or Large Orders, NPL Mathematical Tables 6, Her
                Majesty's Stationery Office, London, 1962.
                N. M. Temme, On the numerical evaluation of the modified
                Bessel function of the third kind, Journal of
                Computational Physics 19, (1975), pp. 324-337.
***ROUTINES CALLED  ASYIK, BESK0, BESK0E, BESK1, BESK1E, BESKNU,
                ILMACH, RLMACH, XERMSG
***REVISION HISTORY  (YYMMDD)
    790201  DATE WRITTEN
    890531  Changed all specific intrinsics to generic.  (WRB)
    890531  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
    900326  Removed duplicate information from DESCRIPTION section.
            (WRB)
    920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE

```

BESK0

```
      FUNCTION BESK0 (X)
***BEGIN PROLOGUE  BESK0
***PURPOSE  Compute the modified (hyperbolic) Bessel function of the
             third kind of order zero.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C10B1
***TYPE      SINGLE PRECISION (BESK0-S, DBESK0-D)
***KEYWORDS  FNLIB, HYPERBOLIC BESSEL FUNCTION,
             MODIFIED BESSEL FUNCTION, ORDER ZERO, SPECIAL FUNCTIONS,
             THIRD KIND
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION
```

BESK0(X) calculates the modified (hyperbolic) Bessel function
of the third kind of order zero for real argument X .GT. 0.0.

Series for BK0	on the interval	0.	to	4.00000D+00
			with weighted error	3.57E-19
			log weighted error	18.45
			significant figures required	17.99
			decimal places required	18.97

```
***REFERENCES  (NONE)
***ROUTINES CALLED  BESJ0, BESK0E, CSEVL, INITS, R1MACH, XERMSG
***REVISION HISTORY  (YMMDD)
    770401  DATE WRITTEN
    890531  Changed all specific intrinsics to generic.  (WRB)
    890531  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
    900326  Removed duplicate information from DESCRIPTION section.
           (WRB)
END PROLOGUE
```

BESK0E

```
      FUNCTION BESK0E (X)
***BEGIN PROLOGUE  BESK0E
***PURPOSE  Compute the exponentially scaled modified (hyperbolic)
            Bessel function of the third kind of order zero.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C10B1
***TYPE      SINGLE PRECISION (BESK0E-S, DBSK0E-D)
***KEYWORDS  EXPONENTIALLY SCALED, FNLIB, HYPERBOLIC BESSEL FUNCTION,
            MODIFIED BESSEL FUNCTION, ORDER ZERO, SPECIAL FUNCTIONS,
            THIRD KIND
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION
```

BESK0E(X) computes the exponentially scaled modified (hyperbolic)
Bessel function of third kind of order zero for real argument
X .GT. 0.0, i.e., EXP(X)*K0(X).

Series for BK0	on the interval	0.	to	4.00000D+00
			with weighted error	3.57E-19
			log weighted error	18.45
			significant figures required	17.99
			decimal places required	18.97

Series for AK0	on the interval	1.25000D-01	to	5.00000D-01
			with weighted error	5.34E-17
			log weighted error	16.27
			significant figures required	14.92
			decimal places required	16.89

Series for AK02	on the interval	0.	to	1.25000D-01
			with weighted error	2.34E-17
			log weighted error	16.63
			significant figures required	14.67
			decimal places required	17.20

```
***REFERENCES  (NONE)
***ROUTINES CALLED  BESJ0, CSEVL, INITS, R1MACH, XERMSG
***REVISION HISTORY  (YYMMDD)
    770401  DATE WRITTEN
    890531  Changed all specific intrinsics to generic.  (WRB)
    890531  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
    900326  Removed duplicate information from DESCRIPTION section.
            (WRB)
END PROLOGUE
```

BESK1

```
      FUNCTION BESK1 (X)
***BEGIN PROLOGUE  BESK1
***PURPOSE  Compute the modified (hyperbolic) Bessel function of the
             third kind of order one.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C10B1
***TYPE      SINGLE PRECISION (BESK1-S, DBESK1-D)
***KEYWORDS  FNLIB, HYPERBOLIC BESSEL FUNCTION,
             MODIFIED BESSEL FUNCTION, ORDER ONE, SPECIAL FUNCTIONS,
             THIRD KIND
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION

      BESK1(X) computes the modified (hyperbolic) Bessel function of third
      kind of order one for real argument X, where X .GT. 0.

      Series for BK1          on the interval  0.          to  4.00000D+00
                                with weighted error  7.02E-18
                                log weighted error  17.15
                                significant figures required  16.73
                                decimal places required  17.67

***REFERENCES  (NONE)
***ROUTINES CALLED  BES11, BESK1E, CSEVL, INITS, R1MACH, XERMSG
***REVISION HISTORY  (YMMDD)
      770401  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890531  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
      900326  Removed duplicate information from DESCRIPTION section.
              (WRB)
      END PROLOGUE
```

BESK1E

```
      FUNCTION BESK1E (X)
***BEGIN PROLOGUE  BESK1E
***PURPOSE  Compute the exponentially scaled modified (hyperbolic)
            Bessel function of the third kind of order one.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C10B1
***TYPE      SINGLE PRECISION (BESK1E-S, DBSK1E-D)
***KEYWORDS  EXPONENTIALLY SCALED, FNLIB, HYPERBOLIC BESSEL FUNCTION,
            MODIFIED BESSEL FUNCTION, ORDER ONE, SPECIAL FUNCTIONS,
            THIRD KIND
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION
```

BESK1E(X) computes the exponentially scaled modified (hyperbolic) Bessel function of third kind of order one for real argument X .GT. 0.0, i.e., $\exp(X)*K_1(X)$.

Series for BK1	on the interval	0.	to	4.00000D+00
			with weighted error	7.02E-18
			log weighted error	17.15
			significant figures required	16.73
			decimal places required	17.67

Series for AK1	on the interval	1.25000D-01	to	5.00000D-01
			with weighted error	6.06E-17
			log weighted error	16.22
			significant figures required	15.41
			decimal places required	16.83

Series for AK12	on the interval	0.	to	1.25000D-01
			with weighted error	2.58E-17
			log weighted error	16.59
			significant figures required	15.22
			decimal places required	17.16

```
***REFERENCES  (NONE)
***ROUTINES CALLED  BES11, CSEVL, INITS, R1MACH, XERMSG
***REVISION HISTORY  (YYMMDD)
    770401  DATE WRITTEN
    890531  Changed all specific intrinsics to generic.  (WRB)
    890531  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
    900326  Removed duplicate information from DESCRIPTION section.
            (WRB)
END PROLOGUE
```

BESKES

```
      SUBROUTINE BESKES (XNU, X, NIN, BKE)
***BEGIN PROLOGUE  BESKES
***PURPOSE  Compute a sequence of exponentially scaled modified Bessel
             functions of the third kind of fractional order.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C10B3
***TYPE      SINGLE PRECISION (BESKES-S, DBSKES-D)
***KEYWORDS  EXPONENTIALLY SCALED, FNLIB, FRACTIONAL ORDER,
             MODIFIED BESSEL FUNCTION, SEQUENCE OF BESSEL FUNCTIONS,
             SPECIAL FUNCTIONS, THIRD KIND
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION

      BESKES computes a sequence of exponentially scaled
      (i.e., multiplied by EXP(X)) modified Bessel
      functions of the third kind of order XNU + I at X, where X .GT. 0,
      XNU lies in (-1,1), and I = 0, 1, ... , NIN - 1, if NIN is positive
      and I = 0, -1, ... , NIN + 1, if NIN is negative.  On return, the
      vector BKE(.) contains the results at X for order starting at XNU.

***REFERENCES  (NONE)
***ROUTINES CALLED  R1MACH, R9KNUS, XERMSG
***REVISION HISTORY  (YYMMDD)
      770601  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890911  Removed unnecessary intrinsics.  (WRB)
      890911  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
      900326  Removed duplicate information from DESCRIPTION section.
              (WRB)
      END PROLOGUE
```

BESKNU

```
SUBROUTINE BESKNU (X, FNU, KODE, N, Y, NZ)
***BEGIN PROLOGUE  BESKNU
***SUBSIDIARY
***PURPOSE  Subsidiary to BESK
***LIBRARY  SLATEC
***TYPE     SINGLE PRECISION (BESKNU-S, DBSKNU-D)
***AUTHOR  Amos, D. E., (SNLA)
***DESCRIPTION
```

Abstract

BESKNU computes N member sequences of K Bessel functions $K/\text{SUB}(FNU+I-1)/(X)$, $I=1,N$ for non-negative orders FNU and positive X. Equations of the references are implemented on small orders DNU for $K/\text{SUB}(DNU)/(X)$ and $K/\text{SUB}(DNU+1)/(X)$. Forward recursion with the three term recursion relation generates higher orders $FNU+I-1$, $I=1,\dots,N$. The parameter KODE permits $K/\text{SUB}(FNU+I-1)/(X)$ values or scaled values $\text{EXP}(X)*K/\text{SUB}(FNU+I-1)/(X)$, $I=1,N$ to be returned.

To start the recursion FNU is normalized to the interval $-0.5.LE.DNU.LT.0.5$. A special form of the power series is implemented on $0.LT.X.LE.X1$ while the Miller algorithm for the K Bessel function in terms of the confluent hypergeometric function $U(FNU+0.5, 2*FNU+1, X)$ is implemented on $X1.LT.X.LE.X2$. For $X.GT.X2$, the asymptotic expansion for large X is used. When FNU is a half odd integer, a special formula for $DNU=-0.5$ and $DNU+1.0=0.5$ is used to start the recursion.

BESKNU assumes that a significant digit $\text{SINH}(X)$ function is available.

Description of Arguments

Input

X	- X.GT.0.0E0
FNU	- Order of initial K function, FNU.GE.0.0E0
N	- Number of members of the sequence, N.GE.1
KODE	- A parameter to indicate the scaling option
	KODE= 1 returns
	$Y(I) = K/\text{SUB}(FNU+I-1)/(X)$
	$I=1,\dots,N$
	= 2 returns
	$Y(I) = \text{EXP}(X)*K/\text{SUB}(FNU+I-1)/(X)$
	$I=1,\dots,N$

Output

Y	- A vector whose first N components contain values for the sequence
	$Y(I) = K/\text{SUB}(FNU+I-1)/(X)$, $I=1,\dots,N$ or
	$Y(I) = \text{EXP}(X)*K/\text{SUB}(FNU+I-1)/(X)$, $I=1,\dots,N$
	depending on KODE
NZ	- Number of components set to zero due to underflow,
	NZ= 0 , Normal return
	NZ.NE.0 , First NZ components of Y set to zero
	due to underflow, $Y(I)=0.0E0$, $I=1,\dots,NZ$

Error Conditions

Improper input arguments - a fatal error

Overflow - a fatal error

Underflow with KODE=1 - a non-fatal error (NZ.NE.0)

***SEE ALSO BESK

***REFERENCES N. M. Temme, On the numerical evaluation of the modified
Bessel function of the third kind, Journal of
Computational Physics 19, (1975), pp. 324-337.

***ROUTINES CALLED GAMMA, ILMACH, RLMACH, XERMSG

***REVISION HISTORY (YYMMDD)

790201 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

891214 Prologue converted to Version 4.0 format. (BAB)

900315 CALLs to XERROR changed to CALLs to XERMSG. (THJ)

900326 Removed duplicate information from DESCRIPTION section.
(WRB)

900328 Added TYPE section. (WRB)

900727 Added EXTERNAL statement. (WRB)

910408 Updated the AUTHOR and REFERENCES sections. (WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

BESKS

```
      SUBROUTINE BESKS (XNU, X, NIN, BK)
***BEGIN PROLOGUE  BESKS
***PURPOSE  Compute a sequence of modified Bessel functions of the
             third kind of fractional order.
***LIBRARY  SLATEC (FNLIB)
***CATEGORY  C10B3
***TYPE      SINGLE PRECISION (BESKS-S, DBESKS-D)
***KEYWORDS  FNLIB, FRACTIONAL ORDER, MODIFIED BESSEL FUNCTION,
             SEQUENCE OF BESSEL FUNCTIONS, SPECIAL FUNCTIONS,
             THIRD KIND
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION

      BESKS computes a sequence of modified Bessel functions of the third
      kind of order XNU + I at X, where X .GT. 0, XNU lies in (-1,1),
      and I = 0, 1, ... , NIN - 1, if NIN is positive and I = 0, 1, ... ,
      NIN + 1, if NIN is negative.  On return, the vector BK(.) Contains
      the results at X for order starting at XNU.

***REFERENCES  (NONE)
***ROUTINES CALLED  BESKES, R1MACH, XERMSG
***REVISION HISTORY  (YYMMDD)
      770601  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890531  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
      900326  Removed duplicate information from DESCRIPTION section.
              (WRB)
      END PROLOGUE
```

BESY

```
SUBROUTINE BESY (X, FNU, N, Y)
***BEGIN PROLOGUE  BESY
***PURPOSE  Implement forward recursion on the three term recursion
            relation for a sequence of non-negative order Bessel
            functions  $Y/\text{SUB}(FNU+I-1)/(X)$ ,  $I=1,\dots,N$  for real, positive
            X and non-negative orders FNU.
***LIBRARY    SLATEC
***CATEGORY   C10A3
***TYPE       SINGLE PRECISION (BESY-S, DBESY-D)
***KEYWORDS   SPECIAL FUNCTIONS, Y BESSEL FUNCTION
***AUTHOR    Amos, D. E., (SNLA)
***DESCRIPTION
```

Abstract

BESY implements forward recursion on the three term recursion relation for a sequence of non-negative order Bessel functions $Y/\text{sub}(FNU+I-1)/(X)$, $I=1,N$ for real X .GT. 0.0E0 and non-negative orders FNU. If FNU .LT. NULIM, orders FNU and FNU+1 are obtained from BESYNU which computes by a power series for X .LE. 2, the K Bessel function of an imaginary argument for 2 .LT. X .LE. 20 and the asymptotic expansion for X .GT. 20.

If FNU .GE. NULIM, the uniform asymptotic expansion is coded in ASYJY for orders FNU and FNU+1 to start the recursion. NULIM is 70 or 100 depending on whether $N=1$ or N .GE. 2. An overflow test is made on the leading term of the asymptotic expansion before any extensive computation is done.

Description of Arguments

Input

X - X .GT. 0.0E0
FNU - order of the initial Y function, FNU .GE. 0.0E0
N - number of members in the sequence, N .GE. 1

Output

Y - a vector whose first N components contain values
 for the sequence $Y(I)=Y/\text{sub}(FNU+I-1)/(X)$, $I=1,N$.

Error Conditions

Improper input arguments - a fatal error
Overflow - a fatal error

```
***REFERENCES  F. W. J. Olver, Tables of Bessel Functions of Moderate
               or Large Orders, NPL Mathematical Tables 6, Her
               Majesty's Stationery Office, London, 1962.
               N. M. Temme, On the numerical evaluation of the modified
               Bessel function of the third kind, Journal of
               Computational Physics 19, (1975), pp. 324-337.
               N. M. Temme, On the numerical evaluation of the ordinary
               Bessel function of the second kind, Journal of
               Computational Physics 21, (1976), pp. 343-350.
***ROUTINES CALLED  ASYJY, BESY0, BESY1, BESYNU, ILMACH, RLMACH,
                  XERMSG, YAIRY
***REVISION HISTORY  (YYMMDD)
```

800501 DATE WRITTEN
890531 Changed all specific intrinsics to generic. (WRB)
890531 REVISION DATE from Version 3.2
891214 Prologue converted to Version 4.0 format. (BAB)
900315 CALLs to XERROR changed to CALLs to XERMSG. (THJ)
900326 Removed duplicate information from DESCRIPTION section.
(WRB)
920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

BESY0

```
      FUNCTION BESY0 (X)
***BEGIN PROLOGUE  BESY0
***PURPOSE  Compute the Bessel function of the second kind of order
            zero.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C10A1
***TYPE      SINGLE PRECISION (BESY0-S, DBESY0-D)
***KEYWORDS  BESSEL FUNCTION, FNLIB, ORDER ZERO, SECOND KIND,
            SPECIAL FUNCTIONS
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION
```

BESY0(X) calculates the Bessel function of the second kind of order zero for real argument X.

```
Series for BY0          on the interval  0.          to 1.60000D+01
                        with weighted error 1.20E-17
                        log weighted error 16.92
                        significant figures required 16.15
                        decimal places required 17.48
```

```
Series for BM0          on the interval  0.          to 6.25000D-02
                        with weighted error 4.98E-17
                        log weighted error 16.30
                        significant figures required 14.97
                        decimal places required 16.96
```

```
Series for BTH0         on the interval  0.          to 6.25000D-02
                        with weighted error 3.67E-17
                        log weighted error 16.44
                        significant figures required 15.53
                        decimal places required 17.13
```

```
***REFERENCES  (NONE)
***ROUTINES CALLED  BESJ0, CSEVL, INITS, R1MACH, XERMSG
***REVISION HISTORY  (YYMMDD)
    770401  DATE WRITTEN
    890531  Changed all specific intrinsics to generic.  (WRB)
    890531  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
    900326  Removed duplicate information from DESCRIPTION section.
            (WRB)
END PROLOGUE
```

BESY1

```
      FUNCTION BESY1 (X)
***BEGIN PROLOGUE  BESY1
***PURPOSE  Compute the Bessel function of the second kind of order
            one.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C10A1
***TYPE      SINGLE PRECISION (BESY1-S, DBESY1-D)
***KEYWORDS  BESSEL FUNCTION, FNLIB, ORDER ONE, SECOND KIND,
            SPECIAL FUNCTIONS
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION
```

BESY1(X) calculates the Bessel function of the second kind of order one for real argument X.

```
Series for BY1          on the interval  0.          to 1.60000D+01
                        with weighted error 1.87E-18
                        log weighted error 17.73
                        significant figures required 17.83
                        decimal places required 18.30
```

```
Series for BM1          on the interval  0.          to 6.25000D-02
                        with weighted error 5.61E-17
                        log weighted error 16.25
                        significant figures required 14.97
                        decimal places required 16.91
```

```
Series for BTH1         on the interval  0.          to 6.25000D-02
                        with weighted error 4.10E-17
                        log weighted error 16.39
                        significant figures required 15.96
                        decimal places required 17.08
```

```
***REFERENCES  (NONE)
***ROUTINES CALLED  BESJ1, CSEVL, INITS, R1MACH, XERMSG
***REVISION HISTORY  (YYMMDD)
    770401  DATE WRITTEN
    890531  Changed all specific intrinsics to generic.  (WRB)
    890531  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
    900326  Removed duplicate information from DESCRIPTION section.
            (WRB)
END PROLOGUE
```

BETA

```
      FUNCTION BETA (A, B)
***BEGIN PROLOGUE  BETA
***PURPOSE  Compute the complete Beta function.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C7B
***TYPE      SINGLE PRECISION (BETA-S, DBETA-D, CBETA-C)
***KEYWORDS  COMPLETE BETA FUNCTION, FNLIB, SPECIAL FUNCTIONS
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION

      BETA computes the complete beta function.

      Input Parameters:
            A    real and positive
            B    real and positive

***REFERENCES  (NONE)
***ROUTINES CALLED  ALBETA, GAMLIM, GAMMA, R1MACH, XERMSG
***REVISION HISTORY  (YMMDD)
      770601  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890531  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
      900326  Removed duplicate information from DESCRIPTION section.
              (WRB)
      900727  Added EXTERNAL statement.  (WRB)
      END PROLOGUE
```

BETAI

```
      REAL FUNCTION BETAI (X, PIN, QIN)
***BEGIN PROLOGUE  BETAI
***PURPOSE  Calculate the incomplete Beta function.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C7F
***TYPE      SINGLE PRECISION (BETAI-S, DBETAI-D)
***KEYWORDS  FNLIB, INCOMPLETE BETA FUNCTION, SPECIAL FUNCTIONS
***AUTHOR   Fullerton, W., (LANL)
***DESCRIPTION

      BETAI calculates the REAL incomplete beta function.

      The incomplete beta function ratio is the probability that a
      random variable from a beta distribution having parameters PIN and
      QIN will be less than or equal to X.

      -- Input Arguments -- All arguments are REAL.
      X          upper limit of integration.  X must be in (0,1) inclusive.
      PIN        first beta distribution parameter.  PIN must be .GT. 0.0.
      QIN        second beta distribution parameter.  QIN must be .GT. 0.0.

***REFERENCES  Nancy E. Bosten and E. L. Battiste, Remark on Algorithm
               179, Communications of the ACM 17, 3 (March 1974),
               pp. 156.
***ROUTINES CALLED  ALBETA, R1MACH, XERMSG
***REVISION HISTORY  (YYMMDD)
      770401  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890531  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
      900326  Removed duplicate information from DESCRIPTION section.
              (WRB)
      920528  DESCRIPTION and REFERENCES sections revised.  (WRB)
END PROLOGUE
```

BFQAD

```
SUBROUTINE BFQAD (F, T, BCOEF, N, K, ID, X1, X2, TOL, QUAD, IERR,  
+ WORK)  
***BEGIN PROLOGUE BFQAD  
***PURPOSE Compute the integral of a product of a function and a  
           derivative of a B-spline.  
***LIBRARY SLATEC  
***CATEGORY H2A2A1, E3, K6  
***TYPE SINGLE PRECISION (BFQAD-S, DBFQAD-D)  
***KEYWORDS INTEGRAL OF B-SPLINE, QUADRATURE  
***AUTHOR Amos, D. E., (SNLA)  
***DESCRIPTION
```

Abstract

BFQAD computes the integral on (X1,X2) of a product of a function F and the ID-th derivative of a K-th order B-spline, using the B-representation (T,BCOEF,N,K). (X1,X2) must be a subinterval of T(K) .LE. X .le. T(N+1). An integration routine BSGQ8 (a modification of GAUS8), integrates the product on sub-intervals of (X1,X2) formed by included (distinct) knots.

Description of Arguments

Input

F - external function of one argument for the integrand $BF(X)=F(X)*BVALU(T,BCOEF,N,K,ID,X,INBV,WORK)$
T - knot array of length N+K
BCOEF - coefficient array of length N
N - length of coefficient array
K - order of B-spline, K .GE. 1
ID - order of the spline derivative, 0 .LE. ID .LE. K-1
ID=0 gives the spline function
X1,X2 - end points of quadrature interval in T(K) .LE. X .LE. T(N+1)
TOL - desired accuracy for the quadrature, suggest $10.*STOL .LT. TOL .LE. 0.1$ where STOL is the single precision unit roundoff for the machine = R1MACH(4)

Output

QUAD - integral of BF(X) on (X1,X2)
IERR - a status code
IERR=1 normal return
2 some quadrature on (X1,X2) does not meet the requested tolerance.
WORK - work vector of length 3*K

Error Conditions

X1 or X2 not in T(K) .LE. X .LE. T(N+1) is a fatal error.
TOL not greater than the single precision unit roundoff or less than 0.1 is a fatal error.
Some quadrature fails to meet the requested tolerance.

```
***REFERENCES D. E. Amos, Quadrature subroutines for splines and  
              B-splines, Report SAND79-1825, Sandia Laboratories,  
              December 1979.  
***ROUTINES CALLED BSGQ8, INTRV, R1MACH, XERMSG
```

***REVISION HISTORY (YYMMDD)
800901 DATE WRITTEN
890531 Changed all specific intrinsics to generic. (WRB)
890531 REVISION DATE from Version 3.2
891214 Prologue converted to Version 4.0 format. (BAB)
900315 CALLs to XERROR changed to CALLs to XERMSG. (THJ)
900326 Removed duplicate information from DESCRIPTION section.
(WRB)
920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

BI

```
      FUNCTION BI (X)
***BEGIN PROLOGUE  BI
***PURPOSE  Evaluate the Bairy function (the Airy function of the
             second kind).
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C10D
***TYPE      SINGLE PRECISION (BI-S, DBI-D)
***KEYWORDS  BAIRY FUNCTION, FNLIB, SPECIAL FUNCTIONS
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION
```

BI(X) calculates the Airy function of the second kind for real argument X.

```
Series for BIF          on the interval -1.00000D+00 to  1.00000D+00
                        with weighted error  1.88E-19
                        log weighted error  18.72
                        significant figures required  17.74
                        decimal places required  19.20
```

```
Series for BIG          on the interval -1.00000D+00 to  1.00000D+00
                        with weighted error  2.61E-17
                        log weighted error  16.58
                        significant figures required  15.17
                        decimal places required  17.03
```

```
Series for BIF2         on the interval  1.00000D+00 to  8.00000D+00
                        with weighted error  1.11E-17
                        log weighted error  16.95
                        approx significant figures required  16.5
                        decimal places required  17.45
```

```
Series for BIG2         on the interval  1.00000D+00 to  8.00000D+00
                        with weighted error  1.19E-18
                        log weighted error  17.92
                        approx significant figures required  17.2
                        decimal places required  18.42
```

```
***REFERENCES  (NONE)
***ROUTINES CALLED  BIE, CSEVL, INITS, R1MACH, R9AIMP, XERMSG
***REVISION HISTORY  (YYMMDD)
  770701  DATE WRITTEN
  890531  Changed all specific intrinsics to generic.  (WRB)
  890531  REVISION DATE from Version 3.2
  891214  Prologue converted to Version 4.0 format.  (BAB)
  900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
  900326  Removed duplicate information from DESCRIPTION section.
          (WRB)
END PROLOGUE
```

BIE

```
      FUNCTION BIE (X)
***BEGIN PROLOGUE  BIE
***PURPOSE  Calculate the Bairy function for a negative argument and an
              exponentially scaled Bairy function for a non-negative
              argument.
***LIBRARY    SLATEC (FNLIB)
***CATEGORY   C10D
***TYPE       SINGLE PRECISION (BIE-S, DBIE-D)
***KEYWORDS   BAIRY FUNCTION, EXPONENTIALLY SCALED, FNLIB,
              SPECIAL FUNCTIONS
***AUTHOR    Fullerton, W., (LANL)
***DESCRIPTION
```

Evaluate BI(X) for X .LE. 0 and BI(X)*EXP(ZETA) where
ZETA = 2/3 * X**(3/2) for X .GE. 0.0

Series for BIF	on the interval -1.00000D+00 to 1.00000D+00
	with weighted error 1.88E-19
	log weighted error 18.72
	significant figures required 17.74
	decimal places required 19.20

Series for BIG	on the interval -1.00000D+00 to 1.00000D+00
	with weighted error 2.61E-17
	log weighted error 16.58
	significant figures required 15.17
	decimal places required 17.03

Series for BIF2	on the interval 1.00000D+00 to 8.00000D+00
	with weighted error 1.11E-17
	log weighted error 16.95
	approx significant figures required 16.5
	decimal places required 17.45

Series for BIG2	on the interval 1.00000D+00 to 8.00000D+00
	with weighted error 1.19E-18
	log weighted error 17.92
	approx significant figures required 17.2
	decimal places required 18.42

Series for BIP	on the interval 1.25000D-01 to 3.53553D-01
	with weighted error 1.91E-17
	log weighted error 16.72
	significant figures required 15.35
	decimal places required 17.41

Series for BIP2	on the interval 0. to 1.25000D-01
	with weighted error 1.05E-18
	log weighted error 17.98
	significant figures required 16.74
	decimal places required 18.71

```
***REFERENCES  (NONE)
***ROUTINES CALLED  CSEVL, INITS, R1MACH, R9AIMP
***REVISION HISTORY  (YYMMDD)
    770701  DATE WRITTEN
```

890206 REVISION DATE from Version 3.2
891214 Prologue converted to Version 4.0 format. (BAB)
END PROLOGUE

BINOM

```
      FUNCTION BINOM (N, M)
***BEGIN PROLOGUE  BINOM
***PURPOSE  Compute the binomial coefficients.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C1
***TYPE      SINGLE PRECISION (BINOM-S, DBINOM-D)
***KEYWORDS  BINOMIAL COEFFICIENTS, FNLIB, SPECIAL FUNCTIONS
***AUTHOR   Fullerton, W., (LANL)
***DESCRIPTION

      BINOM(N,M) calculates the binomial coefficient  $(N!)/((M!)*(N-M)!)$ .

***REFERENCES  (NONE)
***ROUTINES CALLED  ALNREL, R1MACH, R9LGMC, XERMSG
***REVISION HISTORY  (YYMMDD)
      770701  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890531  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
      900326  Removed duplicate information from DESCRIPTION section.
              (WRB)
      END PROLOGUE
```

BINT4

```
SUBROUTINE BINT4 (X, Y, NDATA, IBCL, IBCR, FBCL, FBCR, KNTOPT, T,  
+ BCOEF, N, K, W)  
***BEGIN PROLOGUE BINT4  
***PURPOSE Compute the B-representation of a cubic spline  
           which interpolates given data.  
***LIBRARY SLATEC  
***CATEGORY E1A  
***TYPE SINGLE PRECISION (BINT4-S, DBINT4-D)  
***KEYWORDS B-SPLINE, CUBIC SPLINES, DATA FITTING, INTERPOLATION  
***AUTHOR Amos, D. E., (SNLA)  
***DESCRIPTION
```

Abstract

BINT4 computes the B representation (T,BCOEF,N,K) of a cubic spline (K=4) which interpolates data (X(I)),Y(I)), I=1,NDATA. Parameters IBCL, IBCR, FBCL, FBCR allow the specification of the spline first or second derivative at both X(1) and X(NDATA). When this data is not specified by the problem, it is common practice to use a natural spline by setting second derivatives at X(1) and X(NDATA) to zero (IBCL=IBCR=2,FBCL=FBCR=0.0). The spline is defined on T(4) .LE. X .LE. T(N+1) with (ordered) interior knots at X(I) values where N=NDATA+2. The knots T(1), T(2), T(3) lie to the left of T(4)=X(1) and the knots T(N+2), T(N+3), T(N+4) lie to the right of T(N+1)=X(NDATA) in increasing order. If no extrapolation outside (X(1),X(NDATA)) is anticipated, the knots T(1)=T(2)=T(3)=T(4)=X(1) and T(N+2)=T(N+3)=T(N+4)=T(N+1)=X(NDATA) can be specified by KNTOPT=1. KNTOPT=2 selects a knot placement for T(1), T(2), T(3) to make the first 7 knots symmetric about T(4)=X(1) and similarly for T(N+2), T(N+3), T(N+4) about T(N+1)=X(NDATA). KNTOPT=3 allows the user to make his own selection, in increasing order, for T(1), T(2), T(3) to the left of X(1) and T(N+2), T(N+3), T(N+4) to the right of X(NDATA) in the work array W(1) through W(6). In any case, the interpolation on T(4) .LE. X .LE. T(N+1) by using function BVALU is unique for given boundary conditions.

Description of Arguments

Input

X	- X vector of abscissae of length NDATA, distinct and in increasing order
Y	- Y vector of ordinates of length NDATA
NDATA	- number of data points, NDATA .GE. 2
IBCL	- selection parameter for left boundary condition IBCL = 1 constrain the first derivative at X(1) to FBCL IBCL = 2 constrain the second derivative at X(1) to FBCL
IBCR	- selection parameter for right boundary condition IBCR = 1 constrain first derivative at X(NDATA) to FBCR IBCR = 2 constrain second derivative at X(NDATA) to FBCR
FBCL	- left boundary values governed by IBCL
FBCR	- right boundary values governed by IBCR

KNTOPT - knot selection parameter
 KNTOPT = 1 sets knot multiplicity at T(4) and
 T(N+1) to 4
 = 2 sets a symmetric placement of knots
 about T(4) and T(N+1)
 = 3 sets TNP)=WNP) and T(N+1+I)=w(3+I), I=1,3
 where WNP), I=1,6 is supplied by the user
 W - work array of dimension at least 5*(NDATA+2)
 if KNTOPT=3, then W(1),W(2),W(3) are knot values to
 the left of X(1) and W(4),W(5),W(6) are knot
 values to the right of X(NDATA) in increasing
 order to be supplied by the user

Output

T - knot array of length N+4
 BCOEF - B-spline coefficient array of length N
 N - number of coefficients, N=NDATA+2
 K - order of spline, K=4

Error Conditions

Improper input is a fatal error
 Singular system of equations is a fatal error

***REFERENCES D. E. Amos, Computation with splines and B-splines,
 Report SAND78-1968, Sandia Laboratories, March 1979.
 Carl de Boor, Package for calculating with B-splines,
 SIAM Journal on Numerical Analysis 14, 3 (June 1977),
 pp. 441-472.
 Carl de Boor, A Practical Guide to Splines, Applied
 Mathematics Series 27, Springer-Verlag, New York,
 1978.

***ROUTINES CALLED BNFAC, BNSLV, BSPVD, R1MACH, XERMSG

***REVISION HISTORY (YYMMDD)

800901 DATE WRITTEN
 890531 Changed all specific intrinsics to generic. (WRB)
 890531 REVISION DATE from Version 3.2
 891214 Prologue converted to Version 4.0 format. (BAB)
 900315 CALLs to XERROR changed to CALLs to XERMSG. (THJ)
 900326 Removed duplicate information from DESCRIPTION section.
 (WRB)
 920501 Reformatted the REFERENCES section. (WRB)
 END PROLOGUE

BINTK

```
SUBROUTINE BINTK (X, Y, T, N, K, BCOEF, Q, WORK)
***BEGIN PROLOGUE  BINTK
***PURPOSE  Compute the B-representation of a spline which interpolates
            given data.
***LIBRARY   SLATEC
***CATEGORY  E1A
***TYPE      SINGLE PRECISION (BINTK-S, DBINTK-D)
***KEYWORDS  B-SPLINE, DATA FITTING, INTERPOLATION
***AUTHOR   Amos, D. E., (SNLA)
***DESCRIPTION
```

Written by Carl de Boor and modified by D. E. Amos

Abstract

BINTK is the SPLINT routine of the reference.

BINTK produces the B-spline coefficients, BCOEF, of the B-spline of order K with knots T(I), I=1,...,N+K, which takes on the value Y(I) at X(I), I=1,...,N. The spline or any of its derivatives can be evaluated by calls to BVALU. The I-th equation of the linear system $A \cdot BCOEF = B$ for the coefficients of the interpolant enforces interpolation at X(I), I=1,...,N. Hence, B(I) = Y(I), all I, and A is a band matrix with 2K-1 bands if A is invertible. The matrix A is generated row by row and stored, diagonal by diagonal, in the rows of Q, with the main diagonal going into row K. The banded system is then solved by a call to BNFAC (which constructs the triangular factorization for A and stores it again in Q), followed by a call to BNSLV (which then obtains the solution BCOEF by substitution). BNFAC does no pivoting, since the total positivity of the matrix A makes this unnecessary. The linear system to be solved is (theoretically) invertible if and only if

$$T(I) < X(I) < T(I+K), \quad \text{all } I.$$

Equality is permitted on the left for I=1 and on the right for I=N when K knots are used at X(1) or X(N). Otherwise, violation of this condition is certain to lead to an error.

Description of Arguments

Input

- | | |
|---|--|
| X | - vector of length N containing data point abscissa in strictly increasing order. |
| Y | - corresponding vector of length N containing data point ordinates. |
| T | - knot vector of length N+K
since T(1),...,T(K) ≤ X(1) and T(N+1),...,T(N+K) ≥ X(N), this leaves only N-K knots (not necessarily X(I) values) interior to (X(1),X(N)) |
| N | - number of data points, N ≥ K |
| K | - order of the spline, K ≥ 1 |

Output

- | | |
|-------|---|
| BCOEF | - a vector of length N containing the B-spline coefficients |
| Q | - a work vector of length (2*K-1)*N, containing |

the triangular factorization of the coefficient matrix of the linear system being solved. The coefficients for the interpolant of an additional data set (X(I)),YY(I)), I=1,...,N with the same abscissa can be obtained by loading YY into BCOEF and then executing

CALL BNSLV (Q,2K-1,N,K-1,K-1,BCOEF)

WORK - work vector of length 2*K

Error Conditions

Improper input is a fatal error

Singular system of equations is a fatal error

***REFERENCES D. E. Amos, Computation with splines and B-splines, Report SAND78-1968, Sandia Laboratories, March 1979.
Carl de Boor, Package for calculating with B-splines, SIAM Journal on Numerical Analysis 14, 3 (June 1977), pp. 441-472.
Carl de Boor, A Practical Guide to Splines, Applied Mathematics Series 27, Springer-Verlag, New York, 1978.

***ROUTINES CALLED BNFAC, BNSLV, BSPVN, XERMSG

***REVISION HISTORY (YYMMDD)

800901 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900315 CALLs to XERROR changed to CALLs to XERMSG. (THJ)

900326 Removed duplicate information from DESCRIPTION section. (WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

BISECT

```
      SUBROUTINE BISECT (N, EPS1, D, E, E2, LB, UB, MM, M, W, IND, IERR,  
+      RV4, RV5)  
***BEGIN PROLOGUE  BISECT  
***PURPOSE  Compute the eigenvalues of a symmetric tridiagonal matrix  
            in a given interval using Sturm sequencing.  
***LIBRARY   SLATEC (EISPACK)  
***CATEGORY  D4A5, D4C2A  
***TYPE      SINGLE PRECISION (BISECT-S)  
***KEYWORDS  EIGENVALUES, EISPACK  
***AUTHOR   Smith, B. T., et al.  
***DESCRIPTION
```

This subroutine is a translation of the bisection technique in the ALGOL procedure TRISTURM by Peters and Wilkinson. HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 418-439(1971).

This subroutine finds those eigenvalues of a TRIDIAGONAL SYMMETRIC matrix which lie in a specified interval, using bisection.

On INPUT

N is the order of the matrix. N is an INTEGER variable.

EPS1 is an absolute error tolerance for the computed eigenvalues. If the input EPS1 is non-positive, it is reset for each submatrix to a default value, namely, minus the product of the relative machine precision and the 1-norm of the submatrix. EPS1 is a REAL variable.

D contains the diagonal elements of the input matrix. D is a one-dimensional REAL array, dimensioned D(N).

E contains the subdiagonal elements of the input matrix in its last N-1 positions. E(1) is arbitrary. E is a one-dimensional REAL array, dimensioned E(N).

E2 contains the squares of the corresponding elements of E. E2(1) is arbitrary. E2 is a one-dimensional REAL array, dimensioned E2(N).

LB and UB define the interval to be searched for eigenvalues. If LB is not less than UB, no eigenvalues will be found. LB and UB are REAL variables.

MM should be set to an upper bound for the number of eigenvalues in the interval. WARNING - If more than MM eigenvalues are determined to lie in the interval, an error return is made with no eigenvalues found. MM is an INTEGER variable.

On OUTPUT

EPS1 is unaltered unless it has been reset to its (last) default value.

D and E are unaltered.

Elements of E2, corresponding to elements of E regarded as negligible, have been replaced by zero causing the matrix to split into a direct sum of submatrices. E2(1) is also set to zero.

M is the number of eigenvalues determined to lie in (LB,UB).
M is an INTEGER variable.

W contains the M eigenvalues in ascending order.
W is a one-dimensional REAL array, dimensioned W(MM).

IND contains in its first M positions the submatrix indices associated with the corresponding eigenvalues in W --
1 for eigenvalues belonging to the first submatrix from the top, 2 for those belonging to the second submatrix, etc.
IND is an one-dimensional INTEGER array, dimensioned IND(MM).

IERR is an INTEGER flag set to
Zero for normal return,
3*N+1 if M exceeds MM. In this case, M contains the number of eigenvalues determined to lie in (LB,UB).

RV4 and RV5 are one-dimensional REAL arrays used for temporary storage, dimensioned RV4(N) and RV5(N).

The ALGOL procedure STURMCNT contained in TRISTURM appears in BISECT in-line.

Note that subroutine TQL1 or IMTQL1 is generally faster than BISECT, if more than N/4 eigenvalues are to be found.

Questions and comments should be directed to B. S. Garbow,
Applied Mathematics Division, ARGONNE NATIONAL LABORATORY

***REFERENCES B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow,
Y. Ikebe, V. C. Klema and C. B. Moler, Matrix Eigen-
system Routines - EISPACK Guide, Springer-Verlag,
1976.

***ROUTINES CALLED R1MACH

***REVISION HISTORY (YYMMDD)

760101 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

BLKTRI

```
      SUBROUTINE BLKTRI (IFLG, NP, N, AN, BN, CN, MP, M, AM, BM, CM,  
+      IDIMY, Y, IERROR, W)  
***BEGIN PROLOGUE  BLKTRI  
***PURPOSE  Solve a block tridiagonal system of linear equations  
             (usually resulting from the discretization of separable  
             two-dimensional elliptic equations).  
***LIBRARY   SLATEC (FISHPACK)  
***CATEGORY  I2B4B  
***TYPE      SINGLE PRECISION (BLKTRI-S, CBLKTR-C)  
***KEYWORDS  ELLIPTIC PDE, FISHPACK, TRIDIAGONAL LINEAR SYSTEM  
***AUTHOR   Adams, J., (NCAR)  
             Swarztrauber, P. N., (NCAR)  
             Sweet, R., (NCAR)  
***DESCRIPTION
```

Subroutine BLKTRI Solves a System of Linear Equations of the Form

$$\begin{aligned} &AN(J)*X(I,J-1) + AM(I)*X(I-1,J) + (BN(J)+BM(I))*X(I,J) \\ &+ CN(J)*X(I,J+1) + CM(I)*X(I+1,J) = Y(I,J) \\ &\text{for } I = 1,2,\dots,M \text{ and } J = 1,2,\dots,N. \end{aligned}$$

I+1 and I-1 are evaluated modulo M and J+1 and J-1 modulo N, i.e.,

$$\begin{aligned} X(I,0) &= X(I,N), & X(I,N+1) &= X(I,1), \\ X(0,J) &= X(M,J), & X(M+1,J) &= X(1,J). \end{aligned}$$

These equations usually result from the discretization of separable elliptic equations. Boundary conditions may be Dirichlet, Neumann, or Periodic.

* * * * * ON INPUT * * * * *

IFLG

- = 0 Initialization only. Certain quantities that depend on NP, N, AN, BN, and CN are computed and stored in the work array W.
- = 1 The quantities that were computed in the initialization are used to obtain the solution X(I,J).

NOTE A call with IFLG=0 takes approximately one half the time as a call with IFLG = 1 . However, the initialization does not have to be repeated unless NP, N, AN, BN, or CN change.

NP

- = 0 If AN(1) and CN(N) are not zero, which corresponds to periodic boundary conditions.
- = 1 If AN(1) and CN(N) are zero.

N

The number of unknowns in the J-direction. N must be greater than 4. The operation count is proportional to $MN\log_2(N)$, hence N should be selected less than or equal to M.

AN,BN,CN

One-dimensional arrays of length N that specify the coefficients in the linear equations given above.

MP

= 0 If AM(1) and CM(M) are not zero, which corresponds to periodic boundary conditions.
= 1 If AM(1) = CM(M) = 0 .

M

The number of unknowns in the I-direction. M must be greater than 4.

AM,BM,CM

One-dimensional arrays of length M that specify the coefficients in the linear equations given above.

IDIMY

The row (or first) dimension of the two-dimensional array Y as it appears in the program calling BLKTTRI. This parameter is used to specify the variable dimension of Y. IDIMY must be at least M.

Y

A two-dimensional array that specifies the values of the right side of the linear system of equations given above. Y must be dimensioned at least M*N.

W

A one-dimensional array that must be provided by the user for work space.

If NP=1 define $K = \text{INT}(\log_2(N)) + 1$ and set $L = 2^{**}(K+1)$ then W must have dimension $(K-2)*L + K + 5 + \text{MAX}(2N, 6M)$

If NP=0 define $K = \text{INT}(\log_2(N-1)) + 1$ and set $L = 2^{**}(K+1)$ then W must have dimension $(K-2)*L + K + 5 + 2N + \text{MAX}(2N, 6M)$

****IMPORTANT**** For purposes of checking, the required dimension of W is computed by BLKTTRI and stored in W(1) in floating point format.

* * * * * On Output * * * * *

Y

Contains the solution X.

IERROR

An error flag that indicates invalid input parameters. Except for number zero, a solution is not attempted.

= 0 No error.
= 1 M is less than 5.
= 2 N is less than 5.
= 3 IDIMY is less than M.
= 4 BLKTTRI failed while computing results that depend on the coefficient arrays AN, BN, CN. Check these arrays.
= 5 AN(J)*CN(J-1) is less than 0 for some J. Possible reasons for this condition are
1. The arrays AN and CN are not correct.

2. Too large a grid spacing was used in the discretization of the elliptic equation.
3. The linear equations resulted from a partial differential equation which was not elliptic.

W

Contains intermediate values that must not be destroyed if BLKTRI will be called again with IFLG=1. W(1) contains the number of locations required by W in floating point format.

*Long Description:

* * * * *	Program Specifications	* * * * *
Dimension of Arguments	AN(N),BN(N),CN(N),AM(M),BM(M),CM(M),Y(IDIMY,N) W(See argument list)	
Latest Revision	June 1979	
Required Subprograms	BLKTRI, BLKTRI, PROD, PRODP, CPROD, CPRODP, COMPB, INDXA, INDXB, INDXC, PPADD, PSGF, PPSGF, PPSPF, BSRH, TEVLS, R1MACH	
Special Conditions	The Algorithm may fail if $ABS(BM(I)+BN(J))$ is less than $ABS(AM(I))+ABS(AN(J))+ABS(CM(I))+ABS(CN(J))$ for some I and J. The Algorithm will also fail if $AN(J)*CN(J-1)$ is less than zero for some J. See the description of the output parameter IERROR.	
Common Blocks	CBLKT	
I/O	None	
Precision	Single	
Specialist	Paul Swarztrauber	
Language	FORTRAN	
History	Version 1 September 1973 Version 2 April 1976 Version 3 June 1979	
Algorithm	Generalized Cyclic Reduction (See Reference below)	
Space Required	Control Data 7600	
Portability	American National Standards Institute Fortran. The machine accuracy is set using function R1MACH.	
Required Resident Routines	None	
References	Swarztrauber, P. and R. Sweet, 'Efficient FORTRAN Subprograms For The Solution Of Elliptic Equations' NCAR TN/IA-109, July, 1975, 138 PP. <i>SLATEC2 (AAAAAA through D9UPAK) - 77</i>	

Swarztrauber P. , 'A Direct Method For The Discrete
Solution Of Separable Elliptic Equations', S.I.A.M.
J. Numer. Anal., 11(1974) PP. 1136-1150.

***REFERENCES P. N. Swarztrauber and R. Sweet, Efficient Fortran
subprograms for the solution of elliptic equations,
NCAR TN/IA-109, July 1975, 138 pp.
P. N. Swarztrauber, A direct method for the discrete
solution of separable elliptic equations, SIAM Journal
on Numerical Analysis 11, (1974), pp. 1136-1150.

***ROUTINES CALLED BLKTR1, COMPB, CPROD, CPRODP, PROD, PRODP

***COMMON BLOCKS CBLKT

***REVISION HISTORY (YYMMDD)

801001 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890531 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

BNDACC

```
SUBROUTINE BNDACC (G, MDG, NB, IP, IR, MT, JT)
***BEGIN PROLOGUE  BNDACC
***PURPOSE  Compute the LU factorization of a banded matrices using
            sequential accumulation of rows of the data matrix.
            Exactly one right-hand side vector is permitted.
***LIBRARY    SLATEC
***CATEGORY   D9
***TYPE       SINGLE PRECISION (BNDACC-S, DBNDACC-D)
***KEYWORDS   BANDED MATRIX, CURVE FITTING, LEAST SQUARES
***AUTHOR    Lawson, C. L., (JPL)
            Hanson, R. J., (SNLA)
***DESCRIPTION
```

These subroutines solve the least squares problem $Ax = b$ for banded matrices A using sequential accumulation of rows of the data matrix. Exactly one right-hand side vector is permitted.

These subroutines are intended for the type of least squares systems that arise in applications such as curve or surface fitting of data. The least squares equations are accumulated and processed using only part of the data. This requires a certain user interaction during the solution of $Ax = b$.

Specifically, suppose the data matrix $(A \ B)$ is row partitioned into Q submatrices. Let $(E \ F)$ be the T -th one of these submatrices where $E = (0 \ C \ 0)$. Here the dimension of E is MT by N and the dimension of C is MT by NB . The value of NB is the bandwidth of A . The dimensions of the leading block of zeros in E are MT by $JT-1$.

The user of the subroutine BNDACC provides MT, JT, C and F for $T=1, \dots, Q$. Not all of this data must be supplied at once.

Following the processing of the various blocks $(E \ F)$, the matrix $(A \ B)$ has been transformed to the form $(R \ D)$ where R is upper triangular and banded with bandwidth NB . The least squares system $Rx = d$ is then easily solved using back substitution by executing the statement `CALL BNDSOL(1,...)`. The sequence of values for JT must be nondecreasing. This may require some preliminary interchanges of rows and columns of the matrix A .

The primary reason for these subroutines is that the total processing can take place in a working array of dimension MU by $NB+1$. An acceptable value for MU is

$$MU = \text{MAX}(MT + N + 1),$$

where N is the number of unknowns.

Here the maximum is taken over all values of MT for $T=1, \dots, Q$. Notice that MT can be taken to be as small as one, showing that MU can be as small as $N+2$. The subprogram BNDACC processes the rows more efficiently if MU is large enough so that each new block $(C \ F)$ has a distinct value of JT .

The four principle parts of these algorithms are obtained by the

following call statements

CALL BNDACC(...) Introduce new blocks of data.

CALL BNDSOL(1,...) Compute solution vector and length of residual vector.

CALL BNDSOL(2,...) Given any row vector H solve $YR = H$ for the row vector Y.

CALL BNDSOL(3,...) Given any column vector W solve $RZ = W$ for the column vector Z.

The dots in the above call statements indicate additional arguments that will be specified in the following paragraphs.

The user must dimension the array appearing in the call list..
G(MDG,NB+1)

Description of calling sequence for BNDACC..

The entire set of parameters for BNDACC are

Input..

G(*,*)	The working array into which the user will place the MT by NB+1 block (C F) in rows IR through IR+MT-1, columns 1 through NB+1. See descriptions of IR and MT below.
MDG	The number of rows in the working array G(*,*). The value of MDG should be .GE. MU. The value of MU is defined in the abstract of these subprograms.
NB	The bandwidth of the data matrix A.
IP	Set by the user to the value 1 before the first call to BNDACC. Its subsequent value is controlled by BNDACC to set up for the next call to BNDACC.
IR	Index of the row of G(*,*) where the user is to place the new block of data (C F). Set by the user to the value 1 before the first call to BNDACC. Its subsequent value is controlled by BNDACC. A value of IR .GT. MDG is considered an error.
MT,JT	Set by the user to indicate respectively the number of new rows of data in the block and the index of the first nonzero column in that set of rows (E F) = (0 C 0 F) being processed.

Output..

G(*,*)	The working array which will contain the processed rows of that part of the data matrix which has been passed to BNDACC.
--------	--

IP,IR The values of these arguments are advanced by BNDACC to be ready for storing and processing a new block of data in G(*,*).

Description of calling sequence for BNDSOL..

The user must dimension the arrays appearing in the call list..

G(MDG,NB+1), X(N)

The entire set of parameters for BNDSOL are

Input..

MODE Set by the user to one of the values 1, 2, or 3. These values respectively indicate that the solution of $AX = B$, $YR = H$ or $RZ = W$ is required.

G(*,*),MDG,
NB,IP,IR These arguments all have the same meaning and contents as following the last call to BNDACC.

X(*) With mode=2 or 3 this array contains, respectively, the right-side vectors H or W of the systems $YR = H$ or $RZ = W$.

N The number of variables in the solution vector. If any of the N diagonal terms are zero the subroutine BNDSOL prints an appropriate message. This condition is considered an error.

Output..

X(*) This array contains the solution vectors X, Y or Z of the systems $AX = B$, $YR = H$ or $RZ = W$ depending on the value of MODE=1, 2 or 3.

RNORM If MODE=1 RNORM is the Euclidean length of the residual vector $AX-B$. When MODE=2 or 3 RNORM is set to zero.

Remarks..

To obtain the upper triangular matrix and transformed right-hand side vector D so that the super diagonals of R form the columns of G(*,*), execute the following Fortran statements.

NBP1=NB+1

DO 10 J=1, NBP1

10 G(IR,J) = 0.E0

MT=1

JT=N+1

CALL BNDACC(G,MDG,NB,IP,IR,MT,JT)

***REFERENCES C. L. Lawson and R. J. Hanson, Solving Least Squares
Problems, Prentice-Hall, Inc., 1974, Chapter 27.
***ROUTINES CALLED H12, XERMSG
***REVISION HISTORY (YYMMDD)
790101 DATE WRITTEN
890531 Changed all specific intrinsics to generic. (WRB)
891006 Cosmetic changes to prologue. (WRB)
891006 REVISION DATE from Version 3.2
891214 Prologue converted to Version 4.0 format. (BAB)
900315 CALLs to XERROR changed to CALLs to XERMSG. (THJ)
900326 Removed duplicate information from DESCRIPTION section.
(WRB)
920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

BNDSOL

```
SUBROUTINE BNDSOL (MODE, G, MDG, NB, IP, IR, X, N, RNORM)
***BEGIN PROLOGUE  BNDSOL
***PURPOSE  Solve the least squares problem for a banded matrix using
            sequential accumulation of rows of the data matrix.
            Exactly one right-hand side vector is permitted.
***LIBRARY    SLATEC
***CATEGORY   D9
***TYPE       SINGLE PRECISION (BNDSOL-S, DBNDSL-D)
***KEYWORDS   BANDED MATRIX, CURVE FITTING, LEAST SQUARES
***AUTHOR     Lawson, C. L., (JPL)
              Hanson, R. J., (SNLA)
***DESCRIPTION
```

These subroutines solve the least squares problem $Ax = b$ for banded matrices A using sequential accumulation of rows of the data matrix. Exactly one right-hand side vector is permitted.

These subroutines are intended for the type of least squares systems that arise in applications such as curve or surface fitting of data. The least squares equations are accumulated and processed using only part of the data. This requires a certain user interaction during the solution of $Ax = b$.

Specifically, suppose the data matrix $(A \ B)$ is row partitioned into Q submatrices. Let $(E \ F)$ be the T -th one of these submatrices where $E = (0 \ C \ 0)$. Here the dimension of E is MT by N and the dimension of C is MT by NB . The value of NB is the bandwidth of A . The dimensions of the leading block of zeros in E are MT by $JT-1$.

The user of the subroutine BNDACC provides MT, JT, C and F for $T=1, \dots, Q$. Not all of this data must be supplied at once.

Following the processing of the various blocks $(E \ F)$, the matrix $(A \ B)$ has been transformed to the form $(R \ D)$ where R is upper triangular and banded with bandwidth NB . The least squares system $Rx = d$ is then easily solved using back substitution by executing the statement `CALL BNDSOL(1, ...)`. The sequence of values for JT must be nondecreasing. This may require some preliminary interchanges of rows and columns of the matrix A .

The primary reason for these subroutines is that the total processing can take place in a working array of dimension MU by $NB+1$. An acceptable value for MU is

$$MU = \text{MAX}(MT + N + 1),$$

where N is the number of unknowns.

Here the maximum is taken over all values of MT for $T=1, \dots, Q$. Notice that MT can be taken to be as small as one, showing that MU can be as small as $N+2$. The subprogram BNDACC processes the rows more efficiently if MU is large enough so that each new block $(C \ F)$ has a distinct value of JT .

The four principle parts of these algorithms are obtained by the

following call statements

CALL BNDACC(...) Introduce new blocks of data.

CALL BNDSOL(1,...) Compute solution vector and length of residual vector.

CALL BNDSOL(2,...) Given any row vector H solve $YR = H$ for the row vector Y.

CALL BNDSOL(3,...) Given any column vector W solve $RZ = W$ for the column vector Z.

The dots in the above call statements indicate additional arguments that will be specified in the following paragraphs.

The user must dimension the array appearing in the call list..
G(MDG,NB+1)

Description of calling sequence for BNDACC..

The entire set of parameters for BNDACC are

Input..

G(*,*)	The working array into which the user will place the MT by NB+1 block (C F) in rows IR through IR+MT-1, columns 1 through NB+1. See descriptions of IR and MT below.
MDG	The number of rows in the working array G(*,*). The value of MDG should be .GE. MU. The value of MU is defined in the abstract of these subprograms.
NB	The bandwidth of the data matrix A.
IP	Set by the user to the value 1 before the first call to BNDACC. Its subsequent value is controlled by BNDACC to set up for the next call to BNDACC.
IR	Index of the row of G(*,*) where the user is the user to the value 1 before the first call to BNDACC. Its subsequent value is controlled by BNDACC. A value of IR .GT. MDG is considered an error.
MT,JT	Set by the user to indicate respectively the number of new rows of data in the block and the index of the first nonzero column in that set of rows (E F) = (0 C 0 F) being processed.

Output..

G(*,*)	The working array which will contain the processed rows of that part of the data matrix which has been passed to BNDACC.
IP,IR	The values of these arguments are advanced by BNDACC to be ready for storing and processing

a new block of data in G(*,*).

Description of calling sequence for BNDSOL..

The user must dimension the arrays appearing in the call list..

G(MDG,NB+1), X(N)

The entire set of parameters for BNDSOL are

Input..

MODE	Set by the user to one of the values 1, 2, or 3. These values respectively indicate that the solution of $AX = B$, $YR = H$ or $RZ = W$ is required.
G(*,*),MDG, NB,IP,IR	These arguments all have the same meaning and contents as following the last call to BNDACC.
X(*)	With mode=2 or 3 this array contains, respectively, the right-side vectors H or W of the systems $YR = H$ or $RZ = W$.
N	The number of variables in the solution vector. If any of the N diagonal terms are zero the subroutine BNDSOL prints an appropriate message. This condition is considered an error.

Output..

X(*)	This array contains the solution vectors X, Y or Z of the systems $AX = B$, $YR = H$ or $RZ = W$ depending on the value of MODE=1, 2 or 3.
RNORM	If MODE=1 RNORM is the Euclidean length of the residual vector $AX-B$. When MODE=2 or 3 RNORM is set to zero.

Remarks..

To obtain the upper triangular matrix and transformed right-hand side vector D so that the super diagonals of R form the columns of G(*,*), execute the following Fortran statements.

```
NBP1=NB+1
DO 10 J=1, NBP1
10 G(IR,J) = 0.E0
MT=1
JT=N+1
CALL BNDACC(G,MDG,NB,IP,IR,MT,JT)
```

***REFERENCES C. L. Lawson and R. J. Hanson, Solving Least Squares
SLATEC2 (AAAAAA through D9UPAK) - 85

Problems, Prentice-Hall, Inc., 1974, Chapter 27.

***ROUTINES CALLED XERMSG

***REVISION HISTORY (YYMMDD)

790101 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

891006 Cosmetic changes to prologue. (WRB)

891006 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900315 CALLs to XERROR changed to CALLs to XERMSG. (THJ)

900326 Removed duplicate information from DESCRIPTION section.
(WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

BQR

```
SUBROUTINE BQR (NM, N, MB, A, T, R, IERR, NV, RV)
***BEGIN PROLOGUE  BQR
***PURPOSE  Compute some of the eigenvalues of a real symmetric
            matrix using the QR method with shifts of origin.
***LIBRARY   SLATEC (EISPACK)
***CATEGORY  D4A6
***TYPE      SINGLE PRECISION (BQR-S)
***KEYWORDS  EIGENVALUES, EISPACK
***AUTHOR   Smith, B. T., et al.
***DESCRIPTION
```

This subroutine is a translation of the ALGOL procedure BQR, NUM. MATH. 16, 85-92(1970) by Martin, Reinsch, and Wilkinson. HANDBOOK FOR AUTO. COMP., VOL II-LINEAR ALGEBRA, 266-272(1971).

This subroutine finds the eigenvalue of smallest (usually) magnitude of a REAL SYMMETRIC BAND matrix using the QR algorithm with shifts of origin. Consecutive calls can be made to find further eigenvalues.

On INPUT

NM must be set to the row dimension of the two-dimensional array parameter, A, as declared in the calling program dimension statement. NM is an INTEGER variable.

N is the order of the matrix A. N is an INTEGER variable. N must be less than or equal to NM.

MB is the (half) band width of the matrix, defined as the number of adjacent diagonals, including the principal diagonal, required to specify the non-zero portion of the lower triangle of the matrix. MB is an INTEGER variable. MB must be less than or equal to N on first call.

A contains the lower triangle of the symmetric band input matrix stored as an N by MB array. Its lowest subdiagonal is stored in the last N+1-MB positions of the first column, its next subdiagonal in the last N+2-MB positions of the second column, further subdiagonals similarly, and finally its principal diagonal in the N positions of the last column. Contents of storages not part of the matrix are arbitrary. On a subsequent call, its output contents from the previous call should be passed. A is a two-dimensional REAL array, dimensioned A(NM,MB).

T specifies the shift (of eigenvalues) applied to the diagonal of A in forming the input matrix. What is actually determined is the eigenvalue of A+TI (I is the identity matrix) nearest to T. On a subsequent call, the output value of T from the previous call should be passed if the next nearest eigenvalue is sought. T is a REAL variable.

R should be specified as zero on the first call, and as its output value from the previous call on a subsequent call. It is used to determine when the last row and column of

the transformed band matrix can be regarded as negligible.
R is a REAL variable.

NV must be set to the dimension of the array parameter RV
as declared in the calling program dimension statement.
NV is an INTEGER variable.

On OUTPUT

A contains the transformed band matrix. The matrix A+TI
derived from the output parameters is similar to the
input A+TI to within rounding errors. Its last row and
column are null (if IERR is zero).

T contains the computed eigenvalue of A+TI (if IERR is zero),
where I is the identity matrix.

R contains the maximum of its input value and the norm of the
last column of the input matrix A.

IERR is an INTEGER flag set to
Zero for normal return,
J if the J-th eigenvalue has not been
determined after a total of 30 iterations.

RV is a one-dimensional REAL array of dimension NV which is
at least $(2*MB**2+4*MB-3)$, used for temporary storage. The
first $(3*MB-2)$ locations correspond to the ALGOL array B,
the next $(2*MB-1)$ locations correspond to the ALGOL array H,
and the final $(2*MB**2-MB)$ locations correspond to the MB
by $(2*MB-1)$ ALGOL array U.

NOTE. For a subsequent call, N should be replaced by N-1, but
MB should not be altered even when it exceeds the current N.

Calls PYTHAG(A,B) for $SQRT(A**2 + B**2)$.

Questions and comments should be directed to B. S. Garbow,
Applied Mathematics Division, ARGONNE NATIONAL LABORATORY

***REFERENCES B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow,
Y. Ikebe, V. C. Klema and C. B. Moler, Matrix Eigen-
system Routines - EISPACK Guide, Springer-Verlag,
1976.

***ROUTINES CALLED PYTHAG

***REVISION HISTORY (YYMMDD)

760101 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

BSKIN

```

SUBROUTINE BSKIN (X, N, CODE, M, Y, NZ, IERR)
***BEGIN PROLOGUE  BSKIN
***PURPOSE  Compute repeated integrals of the K-zero Bessel function.
***LIBRARY   SLATEC
***CATEGORY  C10F
***TYPE      SINGLE PRECISION (BSKIN-S, DBSKIN-D)
***KEYWORDS  BICKLEY FUNCTIONS, EXPONENTIAL INTEGRAL,
              INTEGRALS OF BESSEL FUNCTIONS, K-ZERO BESSEL FUNCTION
***AUTHOR  Amos, D. E., (SNLA)
***DESCRIPTION

```

The following definitions are used in BSKIN:

Definition 1

$KI(0,X)$ = K-zero Bessel function.

Definition 2

$KI(N,X)$ = Bickley Function
 $= \int_X^\infty KI(N-1,t)dt$
for $X \geq 0$ and $N = 1, 2, \dots$

BSKIN computes sequences of Bickley functions (repeated integrals of the K_0 Bessel function); i.e. for fixed X and N and $K=1, \dots$, BSKIN computes the M -member sequence

$Y(K) = KI(N+K-1, X)$ for $KODE=1$

or

$Y(K) = \exp(X) * KI(N+K-1, X)$ for $KODE=2$,

for $N \geq 0$ and $X \geq 0$ (N and X cannot be zero simultaneously).

INPUT

X - Argument, $X \geq 0.0E0$
 N - Order of first member of the sequence $N \geq 0$
 $KODE$ - Selection parameter
 $KODE = 1$ returns $Y(K) = KI(N+K-1, X)$, $K=1, M$
 $KODE = 2$ returns $Y(K) = \exp(X) * KI(N+K-1, X)$, $K=1, M$
 M - Number of members in the sequence, $M \geq 1$

OUTPUT

Y - A vector of dimension at least M containing the sequence selected by $KODE$.
 NZ - Underflow flag
 $NZ = 0$ means computation completed
 $NZ = M$ means an exponential underflow occurred on $KODE=1$. $Y(K)=0.0E0$, $K=1, \dots, M$ is returned
 $IERR$ - Error flag
 $IERR = 0$, Normal return, computation completed.
 $IERR = 1$, Input error, no computation.
 $IERR = 2$, Error, no computation. The termination condition was not met.

The nominal computational accuracy is the maximum of unit roundoff ($=R1MACH(4)$) and $1.0e-18$ since critical constants are given to only 18 digits.

DBSKIN is the double precision version of BSKIN.

*Long Description:

Numerical recurrence on

$$(L-1)*KI(L,X) = X(KI(L-3,X) - KI(L-1,X)) + (L-2)*KI(L-2,X)$$

is stable where recurrence is carried forward or backward away from $\text{INT}(X+0.5)$. The power series for indices 0,1 and 2 on $0.le.X.le.2$ starts a stable recurrence for indices greater than 2. If N is sufficiently large ($N.gt.NLIM$), the uniform asymptotic expansion for N to INFINITY is more economical. On $X.gt.2$ the recursion is started by evaluating the uniform expansion for the three members whose indices are closest to $\text{INT}(X+0.5)$ within the set $N,...,N+M-1$. Forward recurrence, backward recurrence or both, complete the sequence depending on the relation of $\text{INT}(X+0.5)$ to the indices $N,...,N+M-1$.

***REFERENCES D. E. Amos, Uniform asymptotic expansions for exponential integrals $E(N,X)$ and Bickley functions $KI(N,X)$, ACM Transactions on Mathematical Software, 1983.

D. E. Amos, A portable Fortran subroutine for the Bickley functions $KI(N,X)$, Algorithm 609, ACM Transactions on Mathematical Software, 1983.

***ROUTINES CALLED BKIAS, BKISR, EXINT, GAMRN, ILMACH, RLMACH

***REVISION HISTORY (YYMMDD)

820601 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

891009 Removed unreferenced statement label. (WRB)

891009 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

BSPDOC

```
SUBROUTINE BSPDOC
***BEGIN PROLOGUE  BSPDOC
***PURPOSE  Documentation for BSPLINE, a package of subprograms for
            working with piecewise polynomial functions
            in B-representation.
***LIBRARY   SLATEC
***CATEGORY  E, E1A, K, Z
***TYPE      ALL (BSPDOC-A)
***KEYWORDS  B-SPLINE, DOCUMENTATION, SPLINES
***AUTHOR    Amos, D. E., (SNLA)
***DESCRIPTION
```

Abstract

BSPDOC is a non-executable, B-spline documentary routine. The narrative describes a B-spline and the routines necessary to manipulate B-splines at a fairly high level. The basic package described herein is that of reference 5 with names altered to prevent duplication and conflicts with routines from reference 3. The call lists used here are also different. Work vectors were added to ensure portability and proper execution in an overlay environment. These work arrays can be used for other purposes except as noted in BSPVN. While most of the original routines in reference 5 were restricted to orders 20 or less, this restriction was removed from all routines except the quadrature routine BSQAD. (See the section below on differentiation and integration for details.)

The subroutines referenced below are single precision routines. Corresponding double precision versions are also part of the package, and these are referenced by prefixing a D in front of the single precision name. For example, BVALU and DBVALU are the single and double precision versions for evaluating a B-spline or any of its derivatives in the B-representation.

****Description of B-Splines****

A collection of polynomials of fixed degree $K-1$ defined on a subdivision $(X(I), X(I+1))$, $I=1, \dots, M-1$ of (A, B) with $X(1)=A$, $X(M)=B$ is called a B-spline of order K . If the spline has $K-2$ continuous derivatives on (A, B) , then the B-spline is simply called a spline of order K . Each of the $M-1$ polynomial pieces has K coefficients, making a total of $K(M-1)$ parameters. This B-spline and its derivatives have $M-2$ jumps at the subdivision points $X(I)$, $I=2, \dots, M-1$. Continuity requirements at these subdivision points add constraints and reduce the number of free parameters. If a B-spline is continuous at each of the $M-2$ subdivision points, there are $K(M-1)-(M-2)$ free parameters; if in addition the B-spline has continuous first derivatives, there are $K(M-1)-2(M-2)$ free parameters, etc., until we get to a spline where we have $K(M-1)-(K-1)(M-2) = M+K-2$ free parameters. Thus, the principle is that increasing the continuity of derivatives decreases the number of free parameters and conversely.

The points at which the polynomials are tied together by the continuity conditions are called knots. If two knots are allowed to come together at some $X(I)$, then we say that we have a knot of multiplicity 2 there, and the knot values are the $X(I)$ value. If we reverse the procedure of the first paragraph, we find that adding a knot to increase multiplicity increases the number of free parameters and, according to the principle above, we thereby introduce a discontinuity in what was the highest continuous derivative at that knot. Thus, the number of free parameters is $N = NU + K - 2$ where NU is the sum of multiplicities at the $X(I)$ values with $X(1)$ and $X(M)$ of multiplicity 1 ($NU = M$ if all knots are simple, i.e., for a spline, all knots have multiplicity 1.) Each knot can have a multiplicity of at most K . A B-spline is commonly written in the B-representation

$$Y(X) = \sum (A(I) * B(I, X), I=1, N)$$

to show the explicit dependence of the spline on the free parameters or coefficients $A(I) = \text{BCOEF}(I)$ and basis functions $B(I, X)$. These basis functions are themselves special B-splines which are zero except on (at most) K adjoining intervals where each $B(I, X)$ is positive and, in most cases, hat or bell-shaped. In order for the nonzero part of $B(1, X)$ to be a spline covering $(X(1), X(2))$, it is necessary to put $K-1$ knots to the left of A and similarly for $B(N, X)$ to the right of B . Thus, the total number of knots for this representation is $NU + 2K - 2 = N + K$. These knots are carried in an array $T(*)$ dimensioned by at least $N + K$. From the construction, $A = T(K)$ and $B = T(N + 1)$ and the spline is defined on $T(K) \text{.LE.} X \text{.LE.} T(N + 1)$. The nonzero part of each basis function lies in the interval $(T(I), T(I + K))$. In many problems where extrapolation beyond A or B is not anticipated, it is common practice to set $T(1) = T(2) = \dots = T(K) = A$ and $T(N + 1) = T(N + 2) = \dots = T(N + K) = B$. In summary, since $T(K)$ and $T(N + 1)$ as well as interior knots can have multiplicity K , the number of free parameters $N = \text{sum of multiplicities} - K$. The fact that each $B(I, X)$ function is nonzero over at most K intervals means that for a given X value, there are at most K nonzero terms of the sum. This leads to banded matrices in linear algebra problems, and references 3 and 6 take advantage of this in constructing higher level routines to achieve speed and avoid ill-conditioning.

****Basic Routines****

The basic routines which most casual users will need are those concerned with direct evaluation of splines or B-splines. Since the B-representation, denoted by (T, BCOEF, N, K) , is preferred because of numerical stability, the knots $T(*)$, the B-spline coefficients $\text{BCOEF}(*)$, the number of coefficients N , and the order K of the polynomial pieces (of degree $K-1$) are usually given. While the knot array runs from $T(1)$ to $T(N + K)$, the B-spline is normally defined on the interval $T(K) \text{.LE.} X \text{.LE.} T(N + 1)$. To evaluate the B-spline or any of its derivatives on this interval, one can use

$$Y = \text{BVALU}(T, \text{BCOEF}, N, K, \text{ID}, X, \text{INBV}, \text{WORK})$$

where ID is an integer for the ID -th derivative, $0 \text{.LE.} \text{ID} \text{.LE.} K-1$. $\text{ID}=0$ gives the zero-th derivative or B-spline value at X .

If $X.LT.T(K)$ or $X.GT.T(N+1)$, whether by mistake or the result of round off accumulation in incrementing X , $BVALU$ gives a diagnostic. $INBV$ is an initialization parameter which is set to 1 on the first call. Distinct splines require distinct $INBV$ parameters. $WORK$ is a scratch vector of length at least $3*K$.

When more conventional communication is needed for publication, physical interpretation, etc., the B-spline coefficients can be converted to piecewise polynomial (PP) coefficients. Thus, the breakpoints (distinct knots) $XI(*)$, the number of polynomial pieces LXI , and the (right) derivatives $C(*,J)$ at each breakpoint $XI(J)$ are needed to define the Taylor expansion to the right of $XI(J)$ on each interval $XI(J).LE$. $X.LT.XI(J+1)$, $J=1,LXI$ where $XI(1)=A$ and $XI(LXI+1)=B$. These are obtained from the $(T,BCOEF,N,K)$ representation by

```
CALL BSPPPP(T,BCOEF,N,K,LDC,C,XI,LXI,WORK)
```

where $LDC \geq K$ is the leading dimension of the matrix C and $WORK$ is a scratch vector of length at least $K*(N+3)$. Then the PP-representation (C,XI,LXI,K) of $Y(X)$, denoted by $Y(J,X)$ on each interval $XI(J).LE.X.LT.XI(J+1)$, is

$$Y(J,X) = \sum (C(I,J)*((X-XI(J))^{*(I-1)})/factorial(I-1), I=1,K)$$

for $J=1,...,LXI$. One must view this conversion from the B- to the PP-representation with some skepticism because the conversion may lose significant digits when the B-spline varies in an almost discontinuous fashion. To evaluate the B-spline or any of its derivatives using the PP-representation, one uses

```
Y = PPVAL(LDC,C,XI,LXI,K,ID,X,INPPV)
```

where ID and $INPPV$ have the same meaning and usage as ID and $INBV$ in $BVALU$.

To determine to what extent the conversion process loses digits, compute the relative error $ABS((Y1-Y2)/Y2)$ over the X interval with $Y1$ from $PPVAL$ and $Y2$ from $BVALU$. A major reason for considering $PPVAL$ is that evaluation is much faster than that from $BVALU$.

Recall that when multiple knots are encountered, jump type discontinuities in the B-spline or its derivatives occur at these knots, and we need to know that $BVALU$ and $PPVAL$ return right limiting values at these knots except at $X=B$ where left limiting values are returned. These values are used for the Taylor expansions about left end points of breakpoint intervals. That is, the derivatives $C(*,J)$ are right derivatives. Note also that a computed X value which, mathematically, would be a knot value may differ from the knot by a round off error. When this happens in evaluating a discontinuous B-spline or some discontinuous derivative, the value at the knot and the value at X can be radically different. In this case, setting X to a T or XI value makes the computation precise. For left limiting values at knots other than $X=B$, see the prologues to $BVALU$ and other routines.

****Interpolation****

BINTK is used to generate B-spline parameters (T,BCOEF,N,K) which will interpolate the data by calls to BVALU. A similar interpolation can also be done for cubic splines using BINT4 or the code in reference 7. If the PP-representation is given, one can evaluate this representation at an appropriate number of abscissas to create data then use BINTK or BINT4 to generate the B-representation.

****Differentiation and Integration****

Derivatives of B-splines are obtained from BVALU or PPVAL. Integrals are obtained from BSQAD using the B-representation (T,BCOEF,N,K) and PPQAD using the PP-representation (C,XI,LXI,K). More complicated integrals involving the product of a function F and some derivative of a B-spline can be evaluated with BFQAD or PFQAD using the B- or PP- representations respectively. All quadrature routines, except for PPQAD, are limited in accuracy to 18 digits or working precision, whichever is smaller. PPQAD is limited to working precision only. In addition, the order K for BSQAD is limited to 20 or less. If orders greater than 20 are required, use BFQAD with $F(X) = 1$.

****Extrapolation****

Extrapolation outside the interval (A,B) can be accomplished easily by the PP-representation using PPVAL. However, caution should be exercised, especially when several knots are located at A or B or when the extrapolation is carried significantly beyond A or B. On the other hand, direct evaluation with BVALU outside $A=T(K).LE.X.LE.T(N+1)=B$ produces an error message, and some manipulation of the knots and coefficients are needed to extrapolate with BVALU. This process is described in reference 6.

****Curve Fitting and Smoothing****

Unless one has many accurate data points, direct interpolation is not recommended for summarizing data. The results are often not in accordance with intuition since the fitted curve tends to oscillate through the set of points. Monotone splines (reference 7) can help curb this undulating tendency but constrained least squares is more likely to give an acceptable fit with fewer parameters. Subroutine FC, described in reference 6, is recommended for this purpose. The output from this fitting process is the B-representation.

**** Routines in the B-Spline Package ****

Single Precision Routines

The subroutines referenced below are SINGLE PRECISION routines. Corresponding DOUBLE PRECISION versions are also part of the package and these are referenced by prefixing a D in front of the single precision name. For example, BVALU and DBVALU are the SINGLE and DOUBLE PRECISION versions for evaluating a B-spline or any of its deriva-

tives in the B-representation.

BINT4 - interpolates with splines of order 4
BINTK - interpolates with splines of order k
BSQAD - integrates the B-representation on subintervals
PPQAD - integrates the PP-representation
BFQAD - integrates the product of a function F and any spline
derivative in the B-representation
PFQAD - integrates the product of a function F and any spline
derivative in the PP-representation
BVALU - evaluates the B-representation or a derivative
PPVAL - evaluates the PP-representation or a derivative
INTRV - gets the largest index of the knot to the left of x
BSPPP - converts from B- to PP-representation
BSPVD - computes nonzero basis functions and derivatives at x
BSPDR - sets up difference array for BSPEV
BSPEV - evaluates the B-representation and derivatives
BSPVN - called by BSPEV, BSPVD, BSPPP and BINTK for function and
derivative evaluations

Auxiliary Routines

BSGQ8,PPGQ8,BNSLV,BNFAC,XERMSG,DBSGQ8,DPPGQ8,DBNSLV,DBNFAC

Machine Dependent Routines

I1MACH, R1MACH, D1MACH

- ***REFERENCES
1. D. E. Amos, Computation with splines and B-splines, Report SAND78-1968, Sandia Laboratories, March 1979.
 2. D. E. Amos, Quadrature subroutines for splines and B-splines, Report SAND79-1825, Sandia Laboratories, December 1979.
 3. Carl de Boor, A Practical Guide to Splines, Applied Mathematics Series 27, Springer-Verlag, New York, 1978.
 4. Carl de Boor, On calculating with B-Splines, Journal of Approximation Theory 6, (1972), pp. 50-62.
 5. Carl de Boor, Package for calculating with B-splines, SIAM Journal on Numerical Analysis 14, 3 (June 1977), pp. 441-472.
 6. R. J. Hanson, Constrained least squares curve fitting to discrete data using B-splines, a users guide, Report SAND78-1291, Sandia Laboratories, December 1978.
 7. F. N. Fritsch and R. E. Carlson, Monotone piecewise cubic interpolation, SIAM Journal on Numerical Analysis 17, 2 (April 1980), pp. 238-246.

***ROUTINES CALLED (NONE)

***REVISION HISTORY (YYMMDD)

810223 DATE WRITTEN

861211 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900723 PURPOSE section revised. (WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

BSPDR

```
SUBROUTINE BSPDR (T, A, N, K, NDERIV, AD)
***BEGIN PROLOGUE  BSPDR
***PURPOSE  Use the B-representation to construct a divided difference
            table preparatory to a (right) derivative calculation.
***LIBRARY  SLATEC
***CATEGORY  E3
***TYPE     SINGLE PRECISION (BSPDR-S, DBSPDR-D)
***KEYWORDS B-SPLINE, DATA FITTING, DIFFERENTIATION OF SPLINES,
            INTERPOLATION
***AUTHOR  Amos, D. E., (SNLA)
***DESCRIPTION
```

Written by Carl de Boor and modified by D. E. Amos

Abstract

BSPDR is the BSPLDR routine of the reference.

BSPDR uses the B-representation (T,A,N,K) to construct a divided difference table ADIF preparatory to a (right) derivative calculation in BSPEV. The lower triangular matrix ADIF is stored in vector AD by columns. The arrays are related by

$$ADIF(I,J) = AD(I-J+1 + (2*N-J+2)*(J-1)/2)$$

$$I = J,N \text{ , } J = 1,NDERIV \text{ .}$$

Description of Arguments

Input

T - knot vector of length N+K
A - B-spline coefficient vector of length N
N - number of B-spline coefficients
 N = sum of knot multiplicities-K
K - order of the spline, K .GE. 1
NDERIV - number of derivatives, 1 .LE. NDERIV .LE. K.
 NDERIV=1 gives the zero-th derivative = function
 value

Output

AD - table of differences in a vector of length
 (2*N-NDERIV+1)*NDERIV/2 for input to BSPEV

Error Conditions

Improper input is a fatal error

```
***REFERENCES  Carl de Boor, Package for calculating with B-splines,
               SIAM Journal on Numerical Analysis 14, 3 (June 1977),
               pp. 441-472.
***ROUTINES CALLED  XERMSG
***REVISION HISTORY  (YMMDD)
   800901  DATE WRITTEN
   890831  Modified array declarations.  (WRB)
   890831  REVISION DATE from Version 3.2
   891214  Prologue converted to Version 4.0 format.  (BAB)
   900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
   900326  Removed duplicate information from DESCRIPTION section.
```

(WRB)
920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

BSPEV

```
SUBROUTINE BSPEV (T, AD, N, K, NDERIV, X, INEV, SVALUE, WORK)
***BEGIN PROLOGUE  BSPEV
***PURPOSE  Calculate the value of the spline and its derivatives from
             the B-representation.
***LIBRARY   SLATEC
***CATEGORY  E3, K6
***TYPE      SINGLE PRECISION (BSPEV-S, DBSPEV-D)
***KEYWORDS  B-SPLINE, DATA FITTING, INTERPOLATION, SPLINES
***AUTHOR   Amos, D. E., (SNLA)
***DESCRIPTION
```

Written by Carl de Boor and modified by D. E. Amos

Abstract

BSPEV is the BSPLEV routine of the reference.

BSPEV calculates the value of the spline and its derivatives at X from the B-representation (T, A, N, K) and returns them in $SVALUE(I), I=1, NDERIV, T(K) \leq X \leq T(N+1)$. $AD(I)$ can be the B-spline coefficients $A(I), I=1, N$ if $NDERIV=1$. Otherwise AD must be computed before hand by a call to BSPDR $(T, A, N, K, NDERIV, AD)$. If $X=T(I), I=K, N$, right limiting values are obtained.

To compute left derivatives or left limiting values at a knot $T(I)$, replace N by $I-1$ and set $X=T(I), I=K+1, N+1$.

BSPEV calls INTRV, BSPVN

Description of Arguments

Input

T	- knot vector of length $N+K$
AD	- vector of length $(2*N-NDERIV+1)*NDERIV/2$ containing the difference table from BSPDR.
N	- number of B-spline coefficients $N = \text{sum of knot multiplicities} - K$
K	- order of the B-spline, $K \geq 1$
NDERIV	- number of derivatives, $1 \leq NDERIV \leq K$. $NDERIV=1$ gives the zero-th derivative = function value
X	- argument, $T(K) \leq X \leq T(N+1)$
INEV	- an initialization parameter which must be set to 1 the first time BSPEV is called.

Output

INEV	- INEV contains information for efficient processing after the initial call and INEV must not be changed by the user. Distinct splines require distinct INEV parameters.
SVALUE	- vector of length $NDERIV$ containing the spline value in $SVALUE(1)$ and the $NDERIV-1$ derivatives in the remaining components.
WORK	- work vector of length $3*K$

Error Conditions

Improper input is a fatal error.

```

***REFERENCES  Carl de Boor, Package for calculating with B-splines,
                SIAM Journal on Numerical Analysis 14, 3 (June 1977),
                pp. 441-472.
***ROUTINES CALLED  BSPVN, INTRV, XERMSG
***REVISION HISTORY  (YYMMDD)
  800901  DATE WRITTEN
  890831  Modified array declarations.  (WRB)
  890831  REVISION DATE from Version 3.2
  891214  Prologue converted to Version 4.0 format.  (BAB)
  900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
  900326  Removed duplicate information from DESCRIPTION section.
          (WRB)
  920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE

```

BSPPP

```
SUBROUTINE BSPPP (T, A, N, K, LDC, C, XI, LXI, WORK)
***BEGIN PROLOGUE  BSPPP
***PURPOSE  Convert the B-representation of a B-spline to the piecewise
            polynomial (PP) form.
***LIBRARY   SLATEC
***CATEGORY  E3, K6
***TYPE      SINGLE PRECISION (BSPPP-S, DBSPPP-D)
***KEYWORDS  B-SPLINE, PIECEWISE POLYNOMIAL
***AUTHOR   Amos, D. E., (SNLA)
***DESCRIPTION
```

Written by Carl de Boor and modified by D. E. Amos

Abstract

BSPPP is the BSPLPP routine of the reference.

BSPPP converts the B-representation (T,A,N,K) to the piecewise polynomial (PP) form (C,XI,LXI,K) for use with PPVAL. Here XI(*), the break point array of length LXI, is the knot array T(*) with multiplicities removed. The columns of the matrix C(I,J) contain the right Taylor derivatives for the polynomial expansion about XI(J) for the intervals XI(J) .LE. X .LE. XI(J+1), I=1,K, J=1,LXI. Function PPVAL makes this evaluation at a specified point X in XI(1) .LE. X .LE. XI(LXI(1) .LE. X .LE. XI+1)

Description of Arguments

Input

T	- knot vector of length N+K
A	- B-spline coefficient vector of length N
N	- number of B-spline coefficients N = sum of knot multiplicities-K
K	- order of the B-spline, K .GE. 1
LDC	- leading dimension of C, LDC .GE. K

Output

C	- matrix of dimension at least (K,LXI) containing right derivatives at break points
XI	- XI break point vector of length LXI+1
LXI	- number of break points, LXI .LE. N-K+1
WORK	- work vector of length K*(N+3)

Error Conditions

Improper input is a fatal error

```
***REFERENCES  Carl de Boor, Package for calculating with B-splines,
               SIAM Journal on Numerical Analysis 14, 3 (June 1977),
               pp. 441-472.
***ROUTINES CALLED  BSPDR, BSPEV, XERMSG
***REVISION HISTORY  (YMMDD)
   800901  DATE WRITTEN
   890831  Modified array declarations.  (WRB)
   890831  REVISION DATE from Version 3.2
   891214  Prologue converted to Version 4.0 format.  (BAB)
   900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
   900326  Removed duplicate information from DESCRIPTION section.
```

(WRB)
920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

BSPPVD

```
SUBROUTINE BSPPVD (T, K, NDERIV, X, ILEFT, LDVNIK, VNIKX, WORK)
***BEGIN PROLOGUE  BSPPVD
***PURPOSE  Calculate the value and all derivatives of order less than
             NDERIV of all basis functions which do not vanish at X.
***LIBRARY   SLATEC
***CATEGORY  E3, K6
***TYPE      SINGLE PRECISION (BSPPVD-S, DBSPPVD-D)
***KEYWORDS  DIFFERENTIATION OF B-SPLINE, EVALUATION OF B-SPLINE
***AUTHOR   Amos, D. E., (SNLA)
***DESCRIPTION
```

Written by Carl de Boor and modified by D. E. Amos

Abstract

BSPPVD is the BSPLVD routine of the reference.

BSPPVD calculates the value and all derivatives of order less than NDERIV of all basis functions which do not (possibly) vanish at X. ILEFT is input such that $T(ILEFT) \leq X < T(ILEFT+1)$. A call to INTRV(T,N+1,X,ILO,ILEFT,MFLAG) will produce the proper ILEFT. The output of BSPPVD is a matrix VNIKX(I,J) of dimension at least (K,NDERIV) whose columns contain the K nonzero basis functions and their NDERIV-1 right derivatives at X, $I=1,K$, $J=1,NDERIV$. These basis functions have indices $ILEFT-K+I$, $I=1,K$, $K \leq ILEFT \leq N$. The nonzero part of the I-th basis function lies in $(T(I),T(I+K))$, $I=1,N$.

If $X=T(ILEFT+1)$ then VNIKX contains left limiting values (left derivatives) at $T(ILEFT+1)$. In particular, $ILEFT = N$ produces left limiting values at the right end point $X=T(N+1)$. To obtain left limiting values at $T(I)$, $I=K+1,N+1$, set $X =$ next lower distinct knot, call INTRV to get ILEFT, set $X=T(I)$, and then call BSPPVD.

Description of Arguments

Input

T - knot vector of length N+K, where
 N = number of B-spline basis functions
 N = sum of knot multiplicities-K
K - order of the B-spline, $K \geq 1$
NDERIV - number of derivatives = NDERIV-1,
 $1 \leq NDERIV \leq K$
X - argument of basis functions,
 $T(K) \leq X \leq T(N+1)$
ILEFT - largest integer such that
 $T(ILEFT) \leq X < T(ILEFT+1)$
LDVNIK - leading dimension of matrix VNIKX

Output

VNIKX - matrix of dimension at least (K,NDERIV) containing the nonzero basis functions at X and their derivatives columnwise.
WORK - a work vector of length $(K+1)*(K+2)/2$

Error Conditions

Improper input is a fatal error

```
***REFERENCES  Carl de Boor, Package for calculating with B-splines,  
                SIAM Journal on Numerical Analysis 14, 3 (June 1977),  
                pp. 441-472.  
***ROUTINES CALLED  BSPVN, XERMSG  
***REVISION HISTORY  (YYMMDD)  
    800901  DATE WRITTEN  
    890531  Changed all specific intrinsics to generic.  (WRB)  
    890831  Modified array declarations.  (WRB)  
    890831  REVISION DATE from Version 3.2  
    891214  Prologue converted to Version 4.0 format.  (BAB)  
    900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)  
    900326  Removed duplicate information from DESCRIPTION section.  
            (WRB)  
    920501  Reformatted the REFERENCES section.  (WRB)  
END PROLOGUE
```

BSPVN

```
      SUBROUTINE BSPVN (T, JHIGH, K, INDEX, X, ILEFT, VNIKX, WORK,  
+      IWORK)  
***BEGIN PROLOGUE  BSPVN  
***PURPOSE  Calculate the value of all (possibly) nonzero basis  
            functions at X.  
***LIBRARY   SLATEC  
***CATEGORY  E3, K6  
***TYPE      SINGLE PRECISION (BSPVN-S, DBSPVN-D)  
***KEYWORDS  EVALUATION OF B-SPLINE  
***AUTHOR   Amos, D. E., (SNLA)  
***DESCRIPTION
```

Written by Carl de Boor and modified by D. E. Amos

Abstract

BSPVN is the BSPLVN routine of the reference.

BSPVN calculates the value of all (possibly) nonzero basis functions at X of order $\text{MAX}(\text{JHIGH}, (\text{J}+1)*(\text{INDEX}-1))$, where $\text{T}(\text{K}) \leq \text{X} \leq \text{T}(\text{N}+1)$ and $\text{J}=\text{IWORK}$ is set inside the routine on the first call when $\text{INDEX}=1$. ILEFT is such that $\text{T}(\text{ILEFT}) \leq \text{X} < \text{T}(\text{ILEFT}+1)$. A call to $\text{INTRV}(\text{T}, \text{N}+1, \text{X}, \text{ILO}, \text{ILEFT}, \text{MFLAG})$ produces the proper ILEFT . BSPVN calculates using the basic algorithm needed in BSPVD. If only basis functions are desired, setting $\text{JHIGH}=\text{K}$ and $\text{INDEX}=1$ can be faster than calling BSPVD, but extra coding is required for derivatives ($\text{INDEX}=2$) and BSPVD is set up for this purpose.

Left limiting values are set up as described in BSPVD.

Description of Arguments

Input

T	- knot vector of length $\text{N}+\text{K}$, where N = number of B-spline basis functions N = sum of knot multiplicities-K
JHIGH	- order of B-spline, $1 \leq \text{JHIGH} \leq \text{K}$
K	- highest possible order
INDEX	- $\text{INDEX} = 1$ gives basis functions of order JHIGH = 2 denotes previous entry with WORK, IWORK values saved for subsequent calls to BSPVN.
X	- argument of basis functions, $\text{T}(\text{K}) \leq \text{X} \leq \text{T}(\text{N}+1)$
ILEFT	- largest integer such that $\text{T}(\text{ILEFT}) \leq \text{X} < \text{T}(\text{ILEFT}+1)$

Output

VNIKX	- vector of length K for spline values.
WORK	- a work vector of length $2*\text{K}$
IWORK	- a work parameter. Both WORK and IWORK contain information necessary to continue for $\text{INDEX} = 2$. When $\text{INDEX} = 1$ exclusively, these are scratch variables and can be used for other purposes.

Error Conditions

Improper input is a fatal error.

```
***REFERENCES  Carl de Boor, Package for calculating with B-splines,
                SIAM Journal on Numerical Analysis 14, 3 (June 1977),
                pp. 441-472.
***ROUTINES CALLED  XERMSG
***REVISION HISTORY  (YYMMDD)
 800901  DATE WRITTEN
 890831  Modified array declarations.  (WRB)
 890831  REVISION DATE from Version 3.2
 891214  Prologue converted to Version 4.0 format.  (BAB)
 900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
 900326  Removed duplicate information from DESCRIPTION section.
        (WRB)
 920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

BSQAD

```
SUBROUTINE BSQAD (T, BCOEF, N, K, X1, X2, BQUAD, WORK)
***BEGIN PROLOGUE  BSQAD
***PURPOSE  Compute the integral of a K-th order B-spline using the
             B-representation.
***LIBRARY   SLATEC
***CATEGORY  H2A2A1, E3, K6
***TYPE      SINGLE PRECISION (BSQAD-S, DBSQAD-D)
***KEYWORDS  INTEGRAL OF B-SPLINES, QUADRATURE
***AUTHOR   Amos, D. E., (SNLA)
***DESCRIPTION
```

Abstract

BSQAD computes the integral on (X1,X2) of a K-th order B-spline using the B-representation (T,BCOEF,N,K). Orders K as high as 20 are permitted by applying a 2, 6, or 10 point Gauss formula on subintervals of (X1,X2) which are formed by included (distinct) knots.

If orders K greater than 20 are needed, use BFQAD with $F(X) = 1$.

Description of Arguments

Input

T - knot array of length N+K
BCOEF - B-spline coefficient array of length N
N - length of coefficient array
K - order of B-spline, 1 .LE. K .LE. 20
X1,X2 - end points of quadrature interval in
 T(K) .LE. X .LE. T(N+1)

Output

BQUAD - integral of the B-spline over (X1,X2)
WORK - work vector of length 3*K

Error Conditions

Improper input is a fatal error

```
***REFERENCES  D. E. Amos, Quadrature subroutines for splines and
                B-splines, Report SAND79-1825, Sandia Laboratories,
                December 1979.
***ROUTINES CALLED  BVALU, INTRV, XERMSG
***REVISION HISTORY  (YMMDD)
800901  DATE WRITTEN
890531  Changed all specific intrinsics to generic.  (WRB)
890531  REVISION DATE from Version 3.2
891214  Prologue converted to Version 4.0 format.  (BAB)
900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
900326  Removed duplicate information from DESCRIPTION section.
        (WRB)
920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

BVALU

```
FUNCTION BVALU (T, A, N, K, IDERIV, X, INBV, WORK)
***BEGIN PROLOGUE  BVALU
***PURPOSE  Evaluate the B-representation of a B-spline at X for the
            function value or any of its derivatives.
***LIBRARY  SLATEC
***CATEGORY  E3, K6
***TYPE     SINGLE PRECISION (BVALU-S, DBVALU-D)
***KEYWORDS  DIFFERENTIATION OF B-SPLINE, EVALUATION OF B-SPLINE
***AUTHOR  Amos, D. E., (SNLA)
***DESCRIPTION
```

Written by Carl de Boor and modified by D. E. Amos

Abstract

BVALU is the BVALUE function of the reference.

BVALU evaluates the B-representation (T,A,N,K) of a B-spline at X for the function value on IDERIV = 0 or any of its derivatives on IDERIV = 1,2,...,K-1. Right limiting values (right derivatives) are returned except at the right end point $X=T(N+1)$ where left limiting values are computed. The spline is defined on $T(K) \leq X \leq T(N+1)$. BVALU returns a fatal error message when X is outside of this interval.

To compute left derivatives or left limiting values at a knot $T(I)$, replace N by I-1 and set $X=T(I)$, $I=K+1,N+1$.

BVALU calls INTRV

Description of Arguments

Input

T	- knot vector of length N+K
A	- B-spline coefficient vector of length N
N	- number of B-spline coefficients N = sum of knot multiplicities-K
K	- order of the B-spline, K ≥ 1
IDERIV	- order of the derivative, 0 \leq IDERIV \leq K-1 IDERIV=0 returns the B-spline value
X	- argument, $T(K) \leq X \leq T(N+1)$
INBV	- an initialization parameter which must be set to 1 the first time BVALU is called.

Output

INBV	- INBV contains information for efficient processing after the initial call and INBV must not be changed by the user. Distinct splines require distinct INBV parameters.
WORK	- work vector of length 3*K.
BVALU	- value of the IDERIV-th derivative at X

Error Conditions

An improper input is a fatal error

```
***REFERENCES  Carl de Boor, Package for calculating with B-splines,
               SIAM Journal on Numerical Analysis 14, 3 (June 1977),
               pp. 441-472.
```

```
***ROUTINES CALLED  INTRV, XERMSG
***REVISION HISTORY  (YYMMDD)
800901  DATE WRITTEN
890531  Changed all specific intrinsics to generic.  (WRB)
890531  REVISION DATE from Version 3.2
891214  Prologue converted to Version 4.0 format.  (BAB)
900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
900326  Removed duplicate information from DESCRIPTION section.
        (WRB)
920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

BVSUP

```
SUBROUTINE BVSUP (Y, NROWY, NCOMP, XPTS, NXPTS, A, NROWA, ALPHA,
  NIC, B, NROWB, BETA, NFC, IGOFX, RE, AE, IFLAG, WORK, NDW,
  + IWORK, NDIW, NEQIVP)
***BEGIN PROLOGUE  BVSUP
***PURPOSE  Solve a linear two-point boundary value problem using
             superposition coupled with an orthonormalization procedure
             and a variable-step integration scheme.
***LIBRARY    SLATEC
***CATEGORY   I1B1
***TYPE       SINGLE PRECISION (BVSUP-S, DBVSUP-D)
***KEYWORDS   ORTHONORMALIZATION, SHOOTING,
             TWO-POINT BOUNDARY VALUE PROBLEM
***AUTHOR    Scott, M. R., (SNLA)
             Watts, H. A., (SNLA)
***DESCRIPTION

*****
Subroutine BVSUP solves a LINEAR two-point boundary-value problem
of the form


$$dY/dX = MATRIX(X,U)*Y(X) + G(X,U)$$


$$A*Y(X_{initial}) = ALPHA, \quad B*Y(X_{final}) = BETA$$


Coupled with the solution of the initial value problem


$$dU/dX = F(X,U)$$


$$U(X_{initial}) = ETA$$


*****
Abstract
The method of solution uses superposition coupled with an
orthonormalization procedure and a variable-step integration
scheme. Each time the superposition solutions start to
lose their numerical linear independence, the vectors are
reorthonormalized before integration proceeds. The underlying
principle of the algorithm is then to piece together the
intermediate (orthogonalized) solutions, defined on the various
subintervals, to obtain the desired solutions.

*****
INPUT to BVSUP
*****

NROWY = Actual row dimension of Y in calling program.
        NROWY must be .GE. NCOMP

NCOMP = Number of components per solution vector.
        NCOMP is equal to number of original differential
        equations.  NCOMP = NIC + NFC.

XPTS = Desired output points for solution. They must be monotonic.
        Xinitial = XPTS(1)
        Xfinal = XPTS(NXPTS)

NXPTS = Number of output points

A(NROWA,NCOMP) = Boundary condition matrix at Xinitial,
```

must be contained in (NIC,NCOMP) sub-matrix.

NROWA = Actual row dimension of A in calling program,
NROWA must be .GE. NIC.

ALPHA(NIC+NEQIVP) = Boundary conditions at Xinitial.
If NEQIVP .GT. 0 (see below), the boundary
conditions at Xinitial for the initial value
equations must be stored starting in
position (NIC + 1) of ALPHA.
Thus, ALPHA(NIC+K) = ETA(K).

NIC = Number of boundary conditions at Xinitial.

B(NROWB,NCOMP) = Boundary condition matrix at Xfinal,
must be contained in (NFC,NCOMP) sub-matrix.

NROWB = Actual row dimension of B in calling program,
NROWB must be .GE. NFC.

BETA(NFC) = Boundary conditions at Xfinal.

NFC = Number of boundary conditions at Xfinal

IGOFX =0 -- The inhomogeneous term G(X) is identically zero.
=1 -- The inhomogeneous term G(X) is not identically zero.
(if IGOFX=1, then subroutine GVEC (or UVEC) must be
supplied).

RE = Relative error tolerance used by the integrator
(see one of the integrators)

AE = Absolute error tolerance used by the integrator
(see one of the integrators)

****NOTE-** RE and AE should not both be zero.

IFLAG = A status parameter used principally for output.
However, for efficient solution of problems which
are originally defined as complex valued (but
converted to real systems to use this code), the
user must set IFLAG=13 on input. See the comment below
for more information on solving such problems.

WORK(NDW) = Floating point array used for internal storage.

NDW = Actual dimension of WORK array allocated by user.
An estimate for NDW can be computed from the following
NDW = 130 + NCOMP**2 * (6 + NXPTS/2 + expected number of
orthonormalizations/8)
For the DISK or TAPE storage mode,
NDW = 6 * NCOMP**2 + 10 * NCOMP + 130
However, when the ADAMS integrator is to be used, the estimates are
NDW = 130 + NCOMP**2 * (13 + NXPTS/2 + expected number of
orthonormalizations/8)
and NDW = 13 * NCOMP**2 + 22 * NCOMP + 130 , respectively.

IWORK(NDIW) = Integer array used for internal storage.

NDIW = Actual dimension of IWORK array allocated by user.
An estimate for NDIW can be computed from the following
SLATEC2 (AAAAAA through D9UPAK) - 110

NDIW = 68 + NCOMP * (1 + expected number of
orthonormalizations)

****NOTE** -- The amount of storage required is problem dependent and may be difficult to predict in advance. Experience has shown that for most problems 20 or fewer orthonormalizations should suffice. If the problem cannot be completed with the allotted storage, then a message will be printed which estimates the amount of storage necessary. In any case, the user can examine the IWORK array for the actual storage requirements, as described in the output information below.

NEQIVP = Number of auxiliary initial value equations being added to the boundary value problem.

****NOTE** -- Occasionally the coefficients MATRIX and/or G may be functions which depend on the independent variable X and on U, the solution of an auxiliary initial value problem. In order to avoid the difficulties associated with interpolation, the auxiliary equations may be solved simultaneously with the given boundary value problem. This initial value problem may be LINEAR or NONLINEAR. See SAND77-1328 for an example.

The user must supply subroutines FMAT, GVEC, UIVP and UVEC, when needed (they MUST be so named), to evaluate the derivatives as follows

A. FMAT must be supplied.

SUBROUTINE FMAT(X,Y,YP)
X = Independent variable (input to FMAT)
Y = Dependent variable vector (input to FMAT)
YP = dY/dX = Derivative vector (output from FMAT)

Compute the derivatives for the HOMOGENEOUS problem
YP(I) = dY(I)/dX = MATRIX(X) * Y(I) , I = 1,...,NCOMP

When (NEQIVP .GT. 0) and MATRIX is dependent on U as well as on X, the following common statement must be included in FMAT

COMMON /MLIVP/ NOFST

For convenience, the U vector is stored at the bottom of the Y array. Thus, during any call to FMAT, U(I) is referenced by Y(NOFST + I).

Subroutine BVDER calls FMAT NFC times to evaluate the homogeneous equations and, if necessary, it calls FMAT once in evaluating the particular solution. Since X remains unchanged in this sequence of calls it is possible to realize considerable computational savings for complicated and expensive evaluations of the MATRIX entries. To do this the user merely passes a variable, say XS, via COMMON where XS is defined in the main program to be any value except the initial X. Then the non-constant elements of MATRIX(X) appearing in the differential equations need only be computed if X is unequal to XS, whereupon XS is reset to X.

B. If NEQIVP .GT. 0 , UIVP must also be supplied.

```

SUBROUTINE UIVP(X,U,UP)
X = Independent variable (input to UIVP)
U = Dependent variable vector (input to UIVP)
UP = dU/dX = Derivative vector (output from UIVP)

```

Compute the derivatives for the auxiliary initial value eqs
 $UP(I) = dU(I)/dX$, $I = 1, \dots, NEQIVP$.

Subroutine BVDER calls UIVP once to evaluate the derivatives for the auxiliary initial value equations.

C. If $NEQIVP = 0$ and $IGOFX = 1$, GVEC must be supplied.

```

SUBROUTINE GVEC(X,G)
X = Independent variable (input to GVEC)
G = Vector of inhomogeneous terms G(X) (output from GVEC)

```

Compute the inhomogeneous terms $G(X)$
 $G(I) = G(X)$ values for $I = 1, \dots, NCOMP$.

Subroutine BVDER calls GVEC in evaluating the particular solution provided $G(X)$ is NOT identically zero. Thus, when $IGOFX=0$, the user need NOT write a GVEC subroutine. Also, the user does not have to bother with the computational savings scheme for GVEC as this is automatically achieved via the BVDER subroutine.

D. If $NEQIVP .GT. 0$ and $IGOFX = 1$, UVEC must be supplied.

```

SUBROUTINE UVEC(X,U,G)
X = Independent variable (input to UVEC)
U = Dependent variable vector from the auxiliary initial
    value problem (input to UVEC)
G = Array of inhomogeneous terms G(X,U)(output from UVEC)

```

Compute the inhomogeneous terms $G(X,U)$
 $G(I) = G(X,U)$ values for $I = 1, \dots, NCOMP$.

Subroutine BVDER calls UVEC in evaluating the particular solution provided $G(X,U)$ is NOT identically zero. Thus, when $IGOFX=0$, the user need NOT write a UVEC subroutine.

The following is optional input to BVSUP to give the user more flexibility in use of the code. See SAND75-0198, SAND77-1328, SAND77-1690, SAND78-0522, and SAND78-1501 for more information.

****CAUTION -- The user MUST zero out IWORK(1),...,IWORK(15) prior to calling BVSUP. These locations define optional input and MUST be zero UNLESS set to special values by the user as described below.

IWORK(1) -- Number of orthonormalization points.
 A value need be set only if IWORK(11) = 1

IWORK(9) -- Integrator and orthonormalization parameter

SLATEC2 (AAAAAA through D9UPAK) - 112

```

        (default value is 1)
        1 = RUNGE-KUTTA-FEHLBERG code using GRAM-SCHMIDT test.
        2 = ADAMS code using GRAM-SCHMIDT TEST.

IWORK(11) -- Orthonormalization points parameter
        (default value is 0)
        0 - Orthonormalization points not pre-assigned.
        1 - Orthonormalization points pre-assigned in
            the first IWORK(1) positions of WORK.

IWORK(12) -- Storage parameter
        (default value is 0)
        0 - All storage IN CORE
        LUN - Homogeneous and inhomogeneous solutions at
            output points and orthonormalization information
            are stored on DISK. The logical unit number to be
            used for DISK I/O (NTAPE) is set to IWORK(12).

WORK(1),... -- Pre-assigned orthonormalization points, stored
            monotonically, corresponding to the direction
            of integration.

```

```

*****
*** COMPLEX VALUED PROBLEM ***
*****

```

****NOTE****

Suppose the original boundary value problem is NC equations of the form

$$dW/dX = MAT(X,U)*W(X) + H(X,U)$$

$$R*W(X_{initial})=GAMMA, \quad S*W(X_{final})=DELTA$$

where all variables are complex valued. The BVSUP code can be used by converting to a real system of size $2*NC$. To solve the larger dimensioned problem efficiently, the user must initialize IFLAG=13 on input and order the vector components according to $Y(1)=real(W(1)), \dots, Y(NC)=real(W(NC)), Y(NC+1)=imag(W(1)), \dots, Y(2*NC)=imag(W(NC))$. Then define

```

MATRIX =
      . real(MAT)      -imag(MAT) .
      .               .
      . imag(MAT)      real(MAT) .
      .               .

```

The matrices A,B and vectors G,ALPHA,BETA must be defined similarly. Further details can be found in SAND78-1501.

```

*****
OUTPUT from BVSUP
*****

```

Y(NROWY,NXPTS) = Solution at specified output points.

IFLAG output values

=-5 Algorithm ,for obtaining starting vectors for the special complex problem structure, was unable to obtain the initial vectors satisfying the necessary independence criteria.

- =-4 Rank of boundary condition matrix A is less than NIC,
as determined by LSSUDS.
- =-2 Invalid input parameters.
- =-1 Insufficient number of storage locations allocated for
WORK or IWORK.

- =0 Indicates successful solution

- =1 A computed solution is returned but UNIQUENESS of the
solution of the boundary-value problem is questionable.
For an eigenvalue problem, this should be treated as a
successful execution since this is the expected mode
of return.
- =2 A computed solution is returned but the EXISTENCE of the
solution to the boundary-value problem is questionable.
- =3 A nontrivial solution approximation is returned although
the boundary condition matrix $B*Y(X_{final})$ is found to be
nonsingular (to the desired accuracy level) while the
right hand side vector is zero. To eliminate this type
of return, the accuracy of the eigenvalue parameter
must be improved.
- ***NOTE- We attempt to diagnose the correct problem behavior
and report possible difficulties by the appropriate
error flag. However, the user should probably resolve
the problem using smaller error tolerances and/or
perturbations in the boundary conditions or other
parameters. This will often reveal the correct
interpretation for the problem posed.

- =13 Maximum number of orthonormalizations attained before
reaching X_{final} .
- =20-flag from integrator (DERKF or DEABM) values can range
from 21 to 25.
- =30 Solution vectors form a dependent set.

WORK(1),...,WORK(IWORK(1)) = Orthonormalization points
determined by BVPOR.

IWORK(1) = Number of orthonormalizations performed by BVPOR.

IWORK(2) = Maximum number of orthonormalizations allowed as
calculated from storage allocated by user.

IWORK(3),IWORK(4),IWORK(5),IWORK(6) Give information about
actual storage requirements for WORK and IWORK
arrays. In particular,
required storage for WORK array is
 $IWORK(3) + IWORK(4)*(expected\ number\ of\ orthonormalizations)$

required storage for IWORK array is
 $IWORK(5) + IWORK(6)*(expected\ number\ of\ orthonormalizations)$

IWORK(8) = Final value of exponent parameter used in tolerance
test for orthonormalization.

IWORK(16) = Number of independent vectors returned from MGSBV.
It is only of interest when IFLAG=30 is obtained.

IWORK(17) = Numerically estimated rank of the boundary
condition matrix defined from $B*Y(X_{final})$

Necessary machine constants are defined in the function routine R1MACH. The user must make sure that the values set in R1MACH are relevant to the computer being used.

- ***REFERENCES M. R. Scott and H. A. Watts, SUPORT - a computer code for two-point boundary-value problems via orthonormalization, SIAM Journal of Numerical Analysis 14, (1977), pp. 40-70.
- B. L. Darlow, M. R. Scott and H. A. Watts, Modifications of SUPORT, a linear boundary value problem solver Part I - pre-assigning orthonormalization points, auxiliary initial value problem, disk or tape storage, Report SAND77-1328, Sandia Laboratories, Albuquerque, New Mexico, 1977.
- B. L. Darlow, M. R. Scott and H. A. Watts, Modifications of SUPORT, a linear boundary value problem solver Part II - inclusion of an Adams integrator, Report SAND77-1690, Sandia Laboratories, Albuquerque, New Mexico, 1977.
- M. E. Lord and H. A. Watts, Modifications of SUPORT, a linear boundary value problem solver Part III - orthonormalization improvements, Report SAND78-0522, Sandia Laboratories, Albuquerque, New Mexico, 1978.
- H. A. Watts, M. R. Scott and M. E. Lord, Computational solution of complex*16 valued boundary problems, Report SAND78-1501, Sandia Laboratories, Albuquerque, New Mexico, 1978.

***ROUTINES CALLED EXBVP, MACON, XERMSG

***COMMON BLOCKS ML15TO, ML17BW, ML18JR, ML5MCO, ML8SZ

***REVISION HISTORY (YYMMDD)

750601 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

890921 Realigned order of variables in certain COMMON blocks. (WRB)

890921 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900510 Convert XERRWV calls to XERMSG calls. (RWC)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

C0LGMC

```
      COMPLEX FUNCTION C0LGMC (Z)
***BEGIN PROLOGUE  C0LGMC
***PURPOSE  Evaluate (Z+0.5)*LOG((Z+1.)/Z) - 1.0 with relative
            accuracy.
***LIBRARY    SLATEC (FNLIB)
***CATEGORY   C7A
***TYPE       COMPLEX (C0LGMC-C)
***KEYWORDS   FNLIB, GAMMA FUNCTION, SPECIAL FUNCTIONS
***AUTHOR    Fullerton, W., (LANL)
***DESCRIPTION

      Evaluate (Z+0.5)*LOG((Z+1.0)/Z) - 1.0  with relative error accuracy
      Let Q = 1.0/Z so that
            (Z+0.5)*LOG(1+1/Z) - 1 = (Z+0.5)*(LOG(1+Q) - Q + Q*Q/2) - Q*Q/4
            = (Z+0.5)*Q**3*C9LN2R(Q) - Q**2/4,
      where C9LN2R is (LOG(1+Q) - Q + 0.5*Q**2) / Q**3.

***REFERENCES  (NONE)
***ROUTINES CALLED  C9LN2R, R1MACH
***REVISION HISTORY  (YYMMDD)
      780401  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890531  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      END PROLOGUE
```

CACOS

```
      COMPLEX FUNCTION CACOS (Z)
***BEGIN PROLOGUE  CACOS
***PURPOSE  Compute the complex arc cosine.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C4A
***TYPE      COMPLEX (CACOS-C)
***KEYWORDS  ARC COSINE, ELEMENTARY FUNCTIONS, FNLIB, TRIGONOMETRIC
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION
```

CACOS(Z) calculates the complex trigonometric arc cosine of Z.
The result is in units of radians, and the real part is in the
first or second quadrant.

```
***REFERENCES  (NONE)
***ROUTINES CALLED  CASIN
***REVISION HISTORY  (YYMMDD)
    770401  DATE WRITTEN
    861211  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    END PROLOGUE
```

CACOSH

```
      COMPLEX FUNCTION CACOSH (Z)
***BEGIN PROLOGUE  CACOSH
***PURPOSE  Compute the arc hyperbolic cosine.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C4C
***TYPE      COMPLEX (ACOSH-S, DACOSH-D, CACOSH-C)
***KEYWORDS  ACOSH, ARC HYPERBOLIC COSINE, ELEMENTARY FUNCTIONS, FNLIB,
              INVERSE HYPERBOLIC COSINE
***AUTHOR   Fullerton, W., (LANL)
***DESCRIPTION
```

CACOSH(Z) calculates the complex arc hyperbolic cosine of Z.

```
***REFERENCES  (NONE)
***ROUTINES CALLED  CACOS
***REVISION HISTORY  (YYMMDD)
      770401  DATE WRITTEN
      861211  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      END PROLOGUE
```

CAIRY

```
SUBROUTINE CAIRY (Z, ID, KODE, AI, NZ, IERR)
***BEGIN PROLOGUE  CAIRY
***PURPOSE  Compute the Airy function Ai(z) or its derivative dAi/dz
             for complex argument z.  A scaling option is available
             to help avoid underflow and overflow.
***LIBRARY    SLATEC
***CATEGORY   C10D
***TYPE       COMPLEX (CAIRY-C, ZAIRY-C)
***KEYWORDS   AIRY FUNCTION, BESSEL FUNCTION OF ORDER ONE THIRD,
             BESSEL FUNCTION OF ORDER TWO THIRDS
***AUTHOR    Amos, D. E., (SNL)
***DESCRIPTION
```

On KODE=1, CAIRY computes the complex Airy function $Ai(z)$ or its derivative dAi/dz on ID=0 or ID=1 respectively. On KODE=2, a scaling option $\exp(zeta)*Ai(z)$ or $\exp(zeta)*dAi/dz$ is provided to remove the exponential decay in $-\pi/3 < \arg(z) < \pi/3$ and the exponential growth in $\pi/3 < \arg(z) < \pi$ where $zeta = (2/3)*z^{3/2}$.

While the Airy functions $Ai(z)$ and dAi/dz are analytic in the whole z -plane, the corresponding scaled functions defined for KODE=2 have a cut along the negative real axis.

Input

Z - Argument of type COMPLEX
ID - Order of derivative, ID=0 or ID=1
KODE - A parameter to indicate the scaling option
 KODE=1 returns
 AI= $Ai(z)$ on ID=0
 AI= dAi/dz on ID=1
 at $z=Z$
 =2 returns
 AI= $\exp(zeta)*Ai(z)$ on ID=0
 AI= $\exp(zeta)*dAi/dz$ on ID=1
 at $z=Z$ where $zeta = (2/3)*z^{3/2}$

Output

AI - Result of type COMPLEX
NZ - Underflow indicator
 NZ=0 Normal return
 NZ=1 AI=0 due to underflow in
 $-\pi/3 < \arg(Z) < \pi/3$ on KODE=1
IERR - Error flag
 IERR=0 Normal return - COMPUTATION COMPLETED
 IERR=1 Input error - NO COMPUTATION
 IERR=2 Overflow - NO COMPUTATION
 (Re(Z) too large with KODE=1)
 IERR=3 Precision warning - COMPUTATION COMPLETED
 (Result has less than half precision)
 IERR=4 Precision error - NO COMPUTATION
 (Result has no precision)
 IERR=5 Algorithmic error - NO COMPUTATION
 (Termination condition not met)

*Long Description:

$Ai(z)$ and dAi/dz are computed from K Bessel functions by

```
Ai(z) = c*sqrt(z)*K(1/3,zeta)
dAi/dz = -c* z *K(2/3,zeta)
c = 1/(pi*sqrt(3))
zeta = (2/3)*z**(3/2)
```

when $abs(z)>1$ and from power series when $abs(z)\leq 1$.

In most complex variable computation, one must evaluate elementary functions. When the magnitude of Z is large, losses of significance by argument reduction occur. Consequently, if the magnitude of $ZETA=(2/3)*Z**(3/2)$ exceeds $U1=\text{SQRT}(0.5/UR)$, then losses exceeding half precision are likely and an error flag $IERR=3$ is triggered where $UR=R1MACH(4)=\text{UNIT ROUND OFF}$. Also, if the magnitude of $ZETA$ is larger than $U2=0.5/UR$, then all significance is lost and $IERR=4$. In order to use the INT function, $ZETA$ must be further restricted not to exceed $U3=11MACH(9)=\text{LARGEST INTEGER}$. Thus, the magnitude of $ZETA$ must be restricted by $\text{MIN}(U2,U3)$. In IEEE arithmetic, $U1,U2$, and $U3$ are approximately $2.0E+3$, $4.2E+6$, $2.1E+9$ in single precision and $4.7E+7$, $2.3E+15$, $2.1E+9$ in double precision. This makes $U2$ limiting in single precision and $U3$ limiting in double precision. This means that the magnitude of Z cannot exceed approximately $3.4E+4$ in single precision and $2.1E+6$ in double precision. This also means that one can expect to retain, in the worst cases on 32-bit machines, no digits in single precision and only 6 digits in double precision.

The approximate relative error in the magnitude of a complex Bessel function can be expressed as $P*10**S$ where $P=\text{MAX}(\text{UNIT ROUND OFF},1.0E-18)$ is the nominal precision and $10**S$ represents the increase in error due to argument reduction in the elementary functions. Here, $S=\text{MAX}(1,\text{ABS}(\text{LOG}_{10}(\text{ABS}(Z))))$, $\text{ABS}(\text{LOG}_{10}(\text{FNU}))$ approximately (i.e., $S=\text{MAX}(1,\text{ABS}(\text{EXPONENT OF ABS}(Z),\text{ABS}(\text{EXPONENT OF FNU})))$). However, the phase angle may have only absolute accuracy. This is most likely to occur when one component (in magnitude) is larger than the other by several orders of magnitude. If one component is $10**K$ larger than the other, then one can expect only $\text{MAX}(\text{ABS}(\text{LOG}_{10}(P))-K, 0)$ significant digits; or, stated another way, when K exceeds the exponent of P , no significant digits remain in the smaller component. However, the phase angle retains absolute accuracy because, in complex arithmetic with precision P , the smaller component will not (as a rule) decrease below P times the magnitude of the larger component. In these extreme cases, the principal phase angle is on the order of $+P$, $-P$, $\text{PI}/2-P$, or $-\text{PI}/2+P$.

- ***REFERENCES
1. M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions, National Bureau of Standards Applied Mathematics Series 55, U. S. Department of Commerce, Tenth Printing (1972) or later.
 2. D. E. Amos, Computation of Bessel Functions of Complex Argument and Large Order, Report SAND83-0643, Sandia National Laboratories, Albuquerque, NM, May 1983.
 3. D. E. Amos, A Subroutine Package for Bessel Functions *SLATEC2 (AAAAAA through D9UPAK) - 120*

- of a Complex Argument and Nonnegative Order, Report SAND85-1018, Sandia National Laboratory, Albuquerque, NM, May 1985.
4. D. E. Amos, A portable package for Bessel functions of a complex argument and nonnegative order, ACM Transactions on Mathematical Software, 12 (September 1986), pp. 265-273.

***ROUTINES CALLED CACAI, CBKNU, ILMACH, RLMACH

***REVISION HISTORY (YYMMDD)

830501 DATE WRITTEN

890801 REVISION DATE from Version 3.2

910415 Prologue converted to Version 4.0 format. (BAB)

920128 Category corrected. (WRB)

920811 Prologue revised. (DWL)

END PROLOGUE

CARG

```
      FUNCTION CARG (Z)
***BEGIN PROLOGUE  CARG
***PURPOSE  Compute the argument of a complex number.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  A4A
***TYPE      COMPLEX (CARG-C)
***KEYWORDS  ARGUMENT OF A COMPLEX NUMBER, ELEMENTARY FUNCTIONS, FNLIB
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION
```

CARG(Z) calculates the argument of the complex number Z. Note that CARG returns a real result. If $Z = X+iY$, then CARG is $\text{ATAN}(Y/X)$, except when both X and Y are zero, in which case the result will be zero.

```
***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
    770401  DATE WRITTEN
    861211  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    END PROLOGUE
```

CASIN

```
      COMPLEX FUNCTION CASIN (ZINP)
***BEGIN PROLOGUE  CASIN
***PURPOSE  Compute the complex arc sine.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C4A
***TYPE      COMPLEX (CASIN-C)
***KEYWORDS  ARC SINE, ELEMENTARY FUNCTIONS, FNLIB, TRIGONOMETRIC
***AUTHOR   Fullerton, W., (LANL)
***DESCRIPTION
```

CASIN(ZINP) calculates the complex trigonometric arc sine of ZINP. The result is in units of radians, and the real part is in the first or fourth quadrant.

```
***REFERENCES  (NONE)
***ROUTINES CALLED  R1MACH
***REVISION HISTORY  (YYMMDD)
    770701  DATE WRITTEN
    890531  Changed all specific intrinsics to generic.  (WRB)
    890531  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    END PROLOGUE
```

CASINH

```
      COMPLEX FUNCTION CASINH (Z)
***BEGIN PROLOGUE  CASINH
***PURPOSE  Compute the arc hyperbolic sine.
***LIBRARY  SLATEC (FNLIB)
***CATEGORY  C4C
***TYPE      COMPLEX (ASINH-S, DASINH-D, CASINH-C)
***KEYWORDS  ARC HYPERBOLIC SINE, ASINH, ELEMENTARY FUNCTIONS, FNLIB,
              INVERSE HYPERBOLIC SINE
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION
```

CASINH(Z) calculates the complex arc hyperbolic sine of Z.

```
***REFERENCES  (NONE)
***ROUTINES CALLED  CASIN
***REVISION HISTORY  (YYMMDD)
      770401  DATE WRITTEN
      861211  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      END PROLOGUE
```

CATAN

```
      COMPLEX FUNCTION CATAN (Z)
***BEGIN PROLOGUE  CATAN
***PURPOSE  Compute the complex arc tangent.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C4A
***TYPE      COMPLEX (CATAN-C)
***KEYWORDS  ARC TANGENT, ELEMENTARY FUNCTIONS, FNLIB, TRIGONOMETRIC
***AUTHOR   Fullerton, W., (LANL)
***DESCRIPTION

      CATAN(Z) calculates the complex trigonometric arc tangent of Z.
      The result is in units of radians, and the real part is in the first
      or fourth quadrant.

***REFERENCES  (NONE)
***ROUTINES CALLED  R1MACH, XERMSG
***REVISION HISTORY  (YYMMDD)
      770801  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890531  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
      900326  Removed duplicate information from DESCRIPTION section.
              (WRB)
      END PROLOGUE
```

CATAN2

```
      COMPLEX FUNCTION CATAN2 (CSN, CCS)
***BEGIN PROLOGUE  CATAN2
***PURPOSE  Compute the complex arc tangent in the proper quadrant.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C4A
***TYPE      COMPLEX (CATAN2-C)
***KEYWORDS  ARC TANGENT, ELEMENTARY FUNCTIONS, FNLIB, POLAR ANGEL,
              QUADRANT, TRIGONOMETRIC
***AUTHOR   Fullerton, W., (LANL)
***DESCRIPTION
```

CATAN2(CSN,CCS) calculates the complex trigonometric arc tangent of the ratio CSN/CCS and returns a result whose real part is in the correct quadrant (within a multiple of 2π). The result is in units of radians and the real part is between $-\pi$ and $+\pi$.

```
***REFERENCES  (NONE)
***ROUTINES CALLED  CATAN, XERMSG
***REVISION HISTORY  (YYMMDD)
    770401  DATE WRITTEN
    890531  Changed all specific intrinsics to generic.  (WRB)
    890531  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
    900326  Removed duplicate information from DESCRIPTION section.
              (WRB)
END PROLOGUE
```

CATANH

```
      COMPLEX FUNCTION CATANH (Z)
***BEGIN PROLOGUE  CATANH
***PURPOSE  Compute the arc hyperbolic tangent.
***LIBRARY  SLATEC (FNLIB)
***CATEGORY  C4C
***TYPE      COMPLEX (ATANH-S, DATANH-D, CATANH-C)
***KEYWORDS  ARC HYPERBOLIC TANGENT, ATANH, ELEMENTARY FUNCTIONS,
              FNLIB, INVERSE HYPERBOLIC TANGENT
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION
```

CATANH(Z) calculates the complex arc hyperbolic tangent of Z.

```
***REFERENCES  (NONE)
***ROUTINES CALLED  CATAN
***REVISION HISTORY  (YYMMDD)
    770401  DATE WRITTEN
    861211  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    END PROLOGUE
```

CAXPY

```
SUBROUTINE CAXPY (N, CA, CX, INCX, CY, INCY)
***BEGIN PROLOGUE  CAXPY
***PURPOSE  Compute a constant times a vector plus a vector.
***LIBRARY   SLATEC (BLAS)
***CATEGORY  D1A7
***TYPE      COMPLEX (SAXPY-S, DAXPY-D, CAXPY-C)
***KEYWORDS  BLAS, LINEAR ALGEBRA, TRIAD, VECTOR
***AUTHOR   Lawson, C. L., (JPL)
            Hanson, R. J., (SNLA)
            Kincaid, D. R., (U. of Texas)
            Krogh, F. T., (JPL)
***DESCRIPTION

        B L A S   Subprogram
Description of Parameters

--Input--
  N  number of elements in input vector(s)
  CA  complex scalar multiplier
  CX  complex vector with N elements
  INCX  storage spacing between elements of CX
  CY  complex vector with N elements
  INCY  storage spacing between elements of CY

--Output--
  CY  complex result (unchanged if N .LE. 0)

Overwrite complex CY with complex  CA*CX + CY.
For I = 0 to N-1, replace  CY(LY+I*INCY) with CA*CX(LX+I*INCX) +
  CY(LY+I*INCY),
where LX = 1 if INCX .GE. 0, else LX = 1+(1-N)*INCX, and LY is
defined in a similar way using INCY.

***REFERENCES  C. L. Lawson, R. J. Hanson, D. R. Kincaid and F. T.
              Krogh, Basic linear algebra subprograms for Fortran
              usage, Algorithm No. 539, Transactions on Mathematical
              Software 5, 3 (September 1979), pp. 308-323.
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
  791001  DATE WRITTEN
  861211  REVISION DATE from Version 3.2
  891214  Prologue converted to Version 4.0 format.  (BAB)
  920310  Corrected definition of LX in DESCRIPTION.  (WRB)
  920501  Reformatted the REFERENCES section.  (WRB)
  920801  Removed variable CANORM.  (RWC, WRB)
END PROLOGUE
```

CBABK2

```
SUBROUTINE CBABK2 (NM, N, LOW, IGH, SCALE, M, ZR, ZI)
***BEGIN PROLOGUE  CBABK2
***PURPOSE  Form the eigenvectors of a complex general matrix from the
             eigenvectors of matrix output from CBAL.
***LIBRARY   SLATEC (EISPACK)
***CATEGORY  D4C4
***TYPE      COMPLEX (BALBAK-S, CBABK2-C)
***KEYWORDS  EIGENVECTORS, EISPACK
***AUTHOR    Smith, B. T., et al.
***DESCRIPTION
```

This subroutine is a translation of the ALGOL procedure CBABK2, which is a complex version of BALBAK, NUM. MATH. 13, 293-304(1969) by Parlett and Reinsch. HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 315-326(1971).

This subroutine forms the eigenvectors of a COMPLEX GENERAL matrix by back transforming those of the corresponding balanced matrix determined by CBAL.

On INPUT

NM must be set to the row dimension of the two-dimensional array parameters, ZR and ZI, as declared in the calling program dimension statement. NM is an INTEGER variable.

N is the order of the matrix $Z=(ZR,ZI)$. N is an INTEGER variable. N must be less than or equal to NM.

LOW and IGH are INTEGER variables determined by CBAL.

SCALE contains information determining the permutations and scaling factors used by CBAL. SCALE is a one-dimensional REAL array, dimensioned SCALE(N).

M is the number of eigenvectors to be back transformed. M is an INTEGER variable.

ZR and ZI contain the real and imaginary parts, respectively, of the eigenvectors to be back transformed in their first M columns. ZR and ZI are two-dimensional REAL arrays, dimensioned ZR(NM,M) and ZI(NM,M).

On OUTPUT

ZR and ZI contain the real and imaginary parts, respectively, of the transformed eigenvectors in their first M columns.

Questions and comments should be directed to B. S. Garbow,
APPLIED MATHEMATICS DIVISION, ARGONNE NATIONAL LABORATORY

```
***REFERENCES  B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow,
               Y. Ikebe, V. C. Klema and C. B. Moler, Matrix Eigen-
               system Routines - EISPACK Guide, Springer-Verlag,
```

1976.
***ROUTINES CALLED (NONE)
***REVISION HISTORY (YYMMDD)
760101 DATE WRITTEN
890831 Modified array declarations. (WRB)
890831 REVISION DATE from Version 3.2
891214 Prologue converted to Version 4.0 format. (BAB)
920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

CBAL

```
SUBROUTINE CBAL (NM, N, AR, AI, LOW, IGH, SCALE)
***BEGIN PROLOGUE  CBAL
***PURPOSE  Balance a complex general matrix and isolate eigenvalues
             whenever possible.
***LIBRARY   SLATEC (EISPACK)
***CATEGORY  D4C1A
***TYPE      COMPLEX (BALANC-S, CBAL-C)
***KEYWORDS  EIGENVECTORS, EISPACK
***AUTHOR    Smith, B. T., et al.
***DESCRIPTION
```

This subroutine is a translation of the ALGOL procedure CBALANCE, which is a complex version of BALANCE, NUM. MATH. 13, 293-304(1969) by Parlett and Reinsch. HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 315-326(1971).

This subroutine balances a COMPLEX matrix and isolates eigenvalues whenever possible.

On INPUT

NM must be set to the row dimension of the two-dimensional array parameters, AR and AI, as declared in the calling program dimension statement. NM is an INTEGER variable.

N is the order of the matrix A=(AR,AI). N is an INTEGER variable. N must be less than or equal to NM.

AR and AI contain the real and imaginary parts, respectively, of the complex matrix to be balanced. AR and AI are two-dimensional REAL arrays, dimensioned AR(NM,N) and AI(NM,N).

On OUTPUT

AR and AI contain the real and imaginary parts, respectively, of the balanced matrix.

LOW and IGH are two INTEGER variables such that AR(I,J) and AI(I,J) are equal to zero if
(1) I is greater than J and
(2) J=1,...,LOW-1 or I=IGH+1,...,N.

SCALE contains information determining the permutations and scaling factors used. SCALE is a one-dimensional REAL array, dimensioned SCALE(N).

Suppose that the principal submatrix in rows LOW through IGH has been balanced, that P(J) denotes the index interchanged with J during the permutation step, and that the elements of the diagonal matrix used are denoted by D(I,J). Then

$$\begin{aligned} \text{SCALE}(J) &= P(J), & \text{for } J &= 1, \dots, \text{LOW}-1 \\ &= D(J,J) & J &= \text{LOW}, \dots, \text{IGH} \\ &= P(J) & J &= \text{IGH}+1, \dots, N. \end{aligned}$$

The order in which the interchanges are made is N to IGH+1, then 1 to LOW-1.

Note that 1 is returned for IGH if IGH is zero formally.

The ALGOL procedure EXC contained in CBALANCE appears in CBAL in line. (Note that the ALGOL roles of identifiers K,L have been reversed.)

Questions and comments should be directed to B. S. Garbow,
APPLIED MATHEMATICS DIVISION, ARGONNE NATIONAL LABORATORY

***REFERENCES B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow,
Y. Ikebe, V. C. Klema and C. B. Moler, Matrix Eigen-
system Routines - EISPACK Guide, Springer-Verlag,
1976.

***ROUTINES CALLED (NONE)

***REVISION HISTORY (YYMMDD)

760101 DATE WRITTEN

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CBESH

```

SUBROUTINE CBESH (Z, FNU, KODE, M, N, CY, NZ, IERR)
***BEGIN PROLOGUE  CBESH
***PURPOSE  Compute a sequence of the Hankel functions  $H(m,a,z)$ 
             for superscript  $m=1$  or  $2$ , real nonnegative orders  $a=b,$ 
              $b+1,\dots$  where  $b>0$ , and nonzero complex argument  $z$ . A
             scaling option is available to help avoid overflow.
***LIBRARY    SLATEC
***CATEGORY   C10A4
***TYPE       COMPLEX (CBESH-C, ZBESH-C)
***KEYWORDS   BESSEL FUNCTIONS OF COMPLEX ARGUMENT,
             BESSEL FUNCTIONS OF THE THIRD KIND, H BESSEL FUNCTIONS,
             HANKEL FUNCTIONS
***AUTHOR    Amos, D. E., (SNL)
***DESCRIPTION

```

On $KODE=1$, CBESH computes an N member sequence of complex Hankel (Bessel) functions $CY(L)=H(M,FNU+L-1,Z)$ for superscript $M=1$ or 2 , real nonnegative orders $FNU+L-1$, $L=1,\dots,N$, and complex nonzero Z in the cut plane $-\pi<\arg(Z)\leq\pi$. On $KODE=2$, CBESH returns the scaled functions

$$CY(L) = H(M,FNU+L-1,Z) \cdot \exp(-(3-2M)Z \cdot i), \quad i^2 = -1$$

which removes the exponential behavior in both the upper and lower half planes. Definitions and notation are found in the NBS Handbook of Mathematical Functions (Ref. 1).

Input

Z - Nonzero argument of type COMPLEX
 FNU - Initial order of type REAL, $FNU \geq 0$
 $KODE$ - A parameter to indicate the scaling option
 $KODE=1$ returns
 $CY(L)=H(M,FNU+L-1,Z)$, $L=1,\dots,N$
 $=2$ returns
 $CY(L)=H(M,FNU+L-1,Z) \cdot \exp(-(3-2M)Z \cdot i)$,
 $L=1,\dots,N$
 M - Superscript of Hankel function, $M=1$ or 2
 N - Number of terms in the sequence, $N \geq 1$

Output

CY - Result vector of type COMPLEX
 NZ - Number of underflows set to zero
 $NZ=0$ Normal return
 $NZ>0$ $CY(L)=0$ for NZ values of L (if $M=1$ and
 $\text{Im}(Z)>0$ or if $M=2$ and $\text{Im}(Z)<0$, then
 $CY(L)=0$ for $L=1,\dots,NZ$; in the complementary half planes, the underflows may not be in an uninterrupted sequence)
 $IERR$ - Error flag
 $IERR=0$ Normal return - COMPUTATION COMPLETED
 $IERR=1$ Input error - NO COMPUTATION
 $IERR=2$ Overflow - NO COMPUTATION
 ($\text{abs}(Z)$ too small and/or $FNU+N-1$
 too large)
 $IERR=3$ Precision warning - COMPUTATION COMPLETED
 (Result has half precision or less)

because $\text{abs}(Z)$ or $\text{FNU}+\text{N}-1$ is large)
 IERR=4 Precision error - NO COMPUTATION
 (Result has no precision because
 $\text{abs}(Z)$ or $\text{FNU}+\text{N}-1$ is too large)
 IERR=5 Algorithmic error - NO COMPUTATION
 (Termination condition not met)

***Long Description:**

The computation is carried out by the formula

$$H(m,a,z) = (1/t) * \exp(-a*t) * K(a, z * \exp(-t))$$

$$t = (3-2*m) * i * \pi / 2$$

where the K Bessel function is computed as described in the prologue to CBESK.

Exponential decay of $H(m,a,z)$ occurs in the upper half z plane for $m=1$ and the lower half z plane for $m=2$. Exponential growth occurs in the complementary half planes. Scaling by $\exp(-(3-2*m)*z*i)$ removes the exponential behavior in the whole z plane as z goes to infinity.

For negative orders, the formula

$$H(m,-a,z) = H(m,a,z) * \exp((3-2*m)*a*\pi*i)$$

can be used.

In most complex variable computation, one must evaluate elementary functions. When the magnitude of Z or $\text{FNU}+\text{N}-1$ is large, losses of significance by argument reduction occur. Consequently, if either one exceeds $U1=\text{SQRT}(0.5/\text{UR})$, then losses exceeding half precision are likely and an error flag IERR=3 is triggered where $\text{UR}=\text{R1MACH}(4)=\text{UNIT ROUND OFF}$. Also, if either is larger than $U2=0.5/\text{UR}$, then all significance is lost and IERR=4. In order to use the INT function, arguments must be further restricted not to exceed the largest machine integer, $U3=\text{I1MACH}(9)$. Thus, the magnitude of Z and $\text{FNU}+\text{N}-1$ is restricted by $\text{MIN}(U2,U3)$. In IEEE arithmetic, $U1, U2$, and $U3$ approximate $2.0\text{E}+3$, $4.2\text{E}+6$, $2.1\text{E}+9$ in single precision and $4.7\text{E}+7$, $2.3\text{E}+15$ and $2.1\text{E}+9$ in double precision. This makes $U2$ limiting in single precision and $U3$ limiting in double precision. This means that one can expect to retain, in the worst cases on IEEE machines, no digits in single precision and only 6 digits in double precision. Similar considerations hold for other machines.

The approximate relative error in the magnitude of a complex Bessel function can be expressed as $P*10^{**S}$ where $P=\text{MAX}(\text{UNIT ROUND OFF}, 1.0\text{E}-18)$ is the nominal precision and 10^{**S} represents the increase in error due to argument reduction in the elementary functions. Here, $S=\text{MAX}(1, \text{ABS}(\text{LOG10}(\text{ABS}(Z))), \text{ABS}(\text{LOG10}(\text{FNU})))$ approximately (i.e., $S=\text{MAX}(1, \text{ABS}(\text{EXPONENT OF } \text{ABS}(Z), \text{ABS}(\text{EXPONENT OF } \text{FNU})))$). However, the phase angle may have only absolute accuracy. This is most likely to occur when one component (in magnitude) is larger than the other by several orders of magnitude. If one component is 10^{**K} larger than the other, then one can expect only $\text{MAX}(\text{ABS}(\text{LOG10}(P))-K, 0)$ significant digits; or, stated another way, when K exceeds

the exponent of P , no significant digits remain in the smaller component. However, the phase angle retains absolute accuracy because, in complex arithmetic with precision P , the smaller component will not (as a rule) decrease below P times the magnitude of the larger component. In these extreme cases, the principal phase angle is on the order of $+P$, $-P$, $\pi/2-P$, or $-\pi/2+P$.

- ***REFERENCES
1. M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions, National Bureau of Standards Applied Mathematics Series 55, U. S. Department of Commerce, Tenth Printing (1972) or later.
 2. D. E. Amos, Computation of Bessel Functions of Complex Argument, Report SAND83-0086, Sandia National Laboratories, Albuquerque, NM, May 1983.
 3. D. E. Amos, Computation of Bessel Functions of Complex Argument and Large Order, Report SAND83-0643, Sandia National Laboratories, Albuquerque, NM, May 1983.
 4. D. E. Amos, A Subroutine Package for Bessel Functions of a Complex Argument and Nonnegative Order, Report SAND85-1018, Sandia National Laboratory, Albuquerque, NM, May 1985.
 5. D. E. Amos, A portable package for Bessel functions of a complex argument and nonnegative order, ACM Transactions on Mathematical Software, 12 (September 1986), pp. 265-273.

***ROUTINES CALLED CACON, CBKNU, CBUNK, CUOIK, ILMACH, RLMACH

***REVISION HISTORY (YYMMDD)

830501 DATE WRITTEN

890801 REVISION DATE from Version 3.2

910415 Prologue converted to Version 4.0 format. (BAB)

920128 Category corrected. (WRB)

920811 Prologue revised. (DWL)

END PROLOGUE

CBESI

```

SUBROUTINE CBESI (Z, FNU, KODE, N, CY, NZ, IERR)
***BEGIN PROLOGUE  CBESI
***PURPOSE  Compute a sequence of the Bessel functions  $I(a,z)$  for
             complex argument  $z$  and real nonnegative orders  $a=b, b+1,$ 
              $b+2, \dots$  where  $b>0$ . A scaling option is available to
             help avoid overflow.
***LIBRARY    SLATEC
***CATEGORY   C10B4
***TYPE       COMPLEX (CBESI-C, ZBESI-C)
***KEYWORDS   BESSEL FUNCTIONS OF COMPLEX ARGUMENT, I BESSEL FUNCTIONS,
             MODIFIED BESSEL FUNCTIONS
***AUTHOR     Amos, D. E., (SNL)
***DESCRIPTION

```

On KODE=1, CBESI computes an N-member sequence of complex Bessel functions $CY(L)=I(FNU+L-1,Z)$ for real nonnegative orders $FNU+L-1$, $L=1, \dots, N$ and complex Z in the cut plane $-\pi < \arg(Z) \leq \pi$. On KODE=2, CBESI returns the scaled functions

$$CY(L) = \exp(-\text{abs}(X)) * I(FNU+L-1, Z), \quad L=1, \dots, N \text{ and } X=\text{Re}(Z)$$

which removes the exponential growth in both the left and right half-planes as Z goes to infinity.

Input

Z - Argument of type COMPLEX
 FNU - Initial order of type REAL, $FNU \geq 0$
 $KODE$ - A parameter to indicate the scaling option
 KODE=1 returns
 $CY(L)=I(FNU+L-1, Z)$, $L=1, \dots, N$
 =2 returns
 $CY(L)=\exp(-\text{abs}(X)) * I(FNU+L-1, Z)$, $L=1, \dots, N$
 where $X=\text{Re}(Z)$
 N - Number of terms in the sequence, $N \geq 1$

Output

CY - Result vector of type COMPLEX
 NZ - Number of underflows set to zero
 $NZ=0$ Normal return
 $NZ>0$ $CY(L)=0$, $L=N-NZ+1, \dots, N$
 $IERR$ - Error flag
 $IERR=0$ Normal return - COMPUTATION COMPLETED
 $IERR=1$ Input error - NO COMPUTATION
 $IERR=2$ Overflow - NO COMPUTATION
 ($\text{Re}(Z)$ too large on KODE=1)
 $IERR=3$ Precision warning - COMPUTATION COMPLETED
 (Result has half precision or less
 because $\text{abs}(Z)$ or $FNU+N-1$ is large)
 $IERR=4$ Precision error - NO COMPUTATION
 (Result has no precision because
 $\text{abs}(Z)$ or $FNU+N-1$ is too large)
 $IERR=5$ Algorithmic error - NO COMPUTATION
 (Termination condition not met)

*Long Description:

The computation of $I(a,z)$ is carried out by the power series for small $\text{abs}(z)$, the asymptotic expansion for large $\text{abs}(z)$, the Miller algorithm normalized by the Wronskian and a Neumann series for intermediate magnitudes of z , and the uniform asymptotic expansions for $I(a,z)$ and $J(a,z)$ for large orders a . Backward recurrence is used to generate sequences or reduce orders when necessary.

The calculations above are done in the right half plane and continued into the left half plane by the formula

$$I(a, z \exp(t)) = \exp(t*a) * I(a, z), \quad \text{Re}(z) > 0 \\ t = i*\pi \text{ or } -i*\pi$$

For negative orders, the formula

$$I(-a, z) = I(a, z) + (2/\pi) * \sin(\pi*a) * K(a, z)$$

can be used. However, for large orders close to integers the function changes radically. When a is a large positive integer, the magnitude of $I(-a, z) = I(a, z)$ is a large negative power of ten. But when a is not an integer, $K(a, z)$ dominates in magnitude with a large positive power of ten and the most that the second term can be reduced is by unit roundoff from the coefficient. Thus, wide changes can occur within unit roundoff of a large integer for a . Here, large means $a > \text{abs}(z)$.

In most complex variable computation, one must evaluate elementary functions. When the magnitude of Z or $\text{FNU} + N - 1$ is large, losses of significance by argument reduction occur. Consequently, if either one exceeds $U1 = \text{SQRT}(0.5/\text{UR})$, then losses exceeding half precision are likely and an error flag $\text{IERR} = 3$ is triggered where $\text{UR} = \text{R1MACH}(4) = \text{UNIT ROUND OFF}$. Also, if either is larger than $U2 = 0.5/\text{UR}$, then all significance is lost and $\text{IERR} = 4$. In order to use the INT function, arguments must be further restricted not to exceed the largest machine integer, $U3 = \text{I1MACH}(9)$. Thus, the magnitude of Z and $\text{FNU} + N - 1$ is restricted by $\text{MIN}(U2, U3)$. In IEEE arithmetic, $U1, U2$, and $U3$ approximate $2.0\text{E}+3$, $4.2\text{E}+6$, $2.1\text{E}+9$ in single precision and $4.7\text{E}+7$, $2.3\text{E}+15$ and $2.1\text{E}+9$ in double precision. This makes $U2$ limiting in single precision and $U3$ limiting in double precision. This means that one can expect to retain, in the worst cases on IEEE machines, no digits in single precision and only 6 digits in double precision. Similar considerations hold for other machines.

The approximate relative error in the magnitude of a complex Bessel function can be expressed as $P * 10^{**S}$ where $P = \text{MAX}(\text{UNIT ROUND OFF}, 1.0\text{E}-18)$ is the nominal precision and 10^{**S} represents the increase in error due to argument reduction in the elementary functions. Here, $S = \text{MAX}(1, \text{ABS}(\text{LOG10}(\text{ABS}(Z))), \text{ABS}(\text{LOG10}(\text{FNU})))$ approximately (i.e., $S = \text{MAX}(1, \text{ABS}(\text{EXPONENT OF } \text{ABS}(Z)), \text{ABS}(\text{EXPONENT OF } \text{FNU}))$). However, the phase angle may have only absolute accuracy. This is most likely to occur when one component (in magnitude) is larger than the other by several orders of magnitude. If one component is 10^{**K} larger than the other, then one can expect only $\text{MAX}(\text{ABS}(\text{LOG10}(P)) - K, 0)$ significant digits; or, stated another way, when K exceeds the exponent of P , no significant digits remain in the smaller

component. However, the phase angle retains absolute accuracy because, in complex arithmetic with precision P , the smaller component will not (as a rule) decrease below P times the magnitude of the larger component. In these extreme cases, the principal phase angle is on the order of $+P$, $-P$, $\pi/2-P$, or $-\pi/2+P$.

- ***REFERENCES
1. M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions, National Bureau of Standards Applied Mathematics Series 55, U. S. Department of Commerce, Tenth Printing (1972) or later.
 2. D. E. Amos, Computation of Bessel Functions of Complex Argument, Report SAND83-0086, Sandia National Laboratories, Albuquerque, NM, May 1983.
 3. D. E. Amos, Computation of Bessel Functions of Complex Argument and Large Order, Report SAND83-0643, Sandia National Laboratories, Albuquerque, NM, May 1983.
 4. D. E. Amos, A Subroutine Package for Bessel Functions of a Complex Argument and Nonnegative Order, Report SAND85-1018, Sandia National Laboratory, Albuquerque, NM, May 1985.
 5. D. E. Amos, A portable package for Bessel functions of a complex argument and nonnegative order, ACM Transactions on Mathematical Software, 12 (September 1986), pp. 265-273.

***ROUTINES CALLED CBINU, ILMACH, RLMACH

***REVISION HISTORY (YYMMDD)

830501 DATE WRITTEN

890801 REVISION DATE from Version 3.2

910415 Prologue converted to Version 4.0 format. (BAB)

920128 Category corrected. (WRB)

920811 Prologue revised. (DWL)

END PROLOGUE

CBESJ

```

SUBROUTINE CBESJ (Z, FNU, KODE, N, CY, NZ, IERR)
***BEGIN PROLOGUE  CBESJ
***PURPOSE  Compute a sequence of the Bessel functions J(a,z) for
             complex argument z and real nonnegative orders a=b,b+1,
             b+2,... where b>0.  A scaling option is available to
             help avoid overflow.
***LIBRARY    SLATEC
***CATEGORY   C10A4
***TYPE       COMPLEX (CBESJ-C, ZBESJ-C)
***KEYWORDS   BESSEL FUNCTIONS OF COMPLEX ARGUMENT,
             BESSEL FUNCTIONS OF THE FIRST KIND, J BESSEL FUNCTIONS
***AUTHOR     Amos, D. E., (SNL)
***DESCRIPTION

```

On KODE=1, CBESJ computes an N member sequence of complex Bessel functions $CY(L)=J(FNU+L-1,Z)$ for real nonnegative orders $FNU+L-1$, $L=1,\dots,N$ and complex Z in the cut plane $-\pi < \arg(Z) \leq \pi$. On KODE=2, CBESJ returns the scaled functions

$$CY(L) = \exp(-\text{abs}(Y)) * J(FNU+L-1, Z), \quad L=1,\dots,N \text{ and } Y=\text{Im}(Z)$$

which remove the exponential growth in both the upper and lower half planes as Z goes to infinity. Definitions and notation are found in the NBS Handbook of Mathematical Functions (Ref. 1).

Input

Z - Argument of type COMPLEX
 FNU - Initial order of type REAL, $FNU \geq 0$
 $KODE$ - A parameter to indicate the scaling option
 KODE=1 returns
 $CY(L)=J(FNU+L-1,Z)$, $L=1,\dots,N$
 =2 returns
 $CY(L)=J(FNU+L-1,Z)*\exp(-\text{abs}(Y))$, $L=1,\dots,N$
 where $Y=\text{Im}(Z)$
 N - Number of terms in the sequence, $N \geq 1$

Output

CY - Result vector of type COMPLEX
 NZ - Number of underflows set to zero
 $NZ=0$ Normal return
 $NZ>0$ $CY(L)=0$, $L=N-NZ+1,\dots,N$
 $IERR$ - Error flag
 $IERR=0$ Normal return - COMPUTATION COMPLETED
 $IERR=1$ Input error - NO COMPUTATION
 $IERR=2$ Overflow - NO COMPUTATION
 ($\text{Im}(Z)$ too large on KODE=1)
 $IERR=3$ Precision warning - COMPUTATION COMPLETED
 (Result has half precision or less
 because $\text{abs}(Z)$ or $FNU+N-1$ is large)
 $IERR=4$ Precision error - NO COMPUTATION
 (Result has no precision because
 $\text{abs}(Z)$ or $FNU+N-1$ is too large)
 $IERR=5$ Algorithmic error - NO COMPUTATION
 (Termination condition not met)

*Long Description:

The computation is carried out by the formulae

$$J(a,z) = \exp(a\pi i/2) I(a, -i z), \quad \text{Im}(z) \geq 0$$

$$J(a,z) = \exp(-a\pi i/2) I(a, i z), \quad \text{Im}(z) < 0$$

where the I Bessel function is computed as described in the prologue to CBESI.

For negative orders, the formula

$$J(-a,z) = J(a,z) \cos(a\pi) - Y(a,z) \sin(a\pi)$$

can be used. However, for large orders close to integers, the function changes radically. When a is a large positive integer, the magnitude of $J(-a,z) = J(a,z) \cos(a\pi)$ is a large negative power of ten. But when a is not an integer, $Y(a,z)$ dominates in magnitude with a large positive power of ten and the most that the second term can be reduced is by unit roundoff from the coefficient. Thus, wide changes can occur within unit roundoff of a large integer for a . Here, large means $a > \text{abs}(z)$.

In most complex variable computation, one must evaluate elementary functions. When the magnitude of Z or $\text{FNU} + N - 1$ is large, losses of significance by argument reduction occur. Consequently, if either one exceeds $U1 = \text{SQRT}(0.5/\text{UR})$, then losses exceeding half precision are likely and an error flag $\text{IERR} = 3$ is triggered where $\text{UR} = \text{R1MACH}(4) = \text{UNIT ROUNDOFF}$. Also, if either is larger than $U2 = 0.5/\text{UR}$, then all significance is lost and $\text{IERR} = 4$. In order to use the INT function, arguments must be further restricted not to exceed the largest machine integer, $U3 = \text{I1MACH}(9)$. Thus, the magnitude of Z and $\text{FNU} + N - 1$ is restricted by $\text{MIN}(U2, U3)$. In IEEE arithmetic, $U1, U2$, and $U3$ approximate $2.0\text{E}+3$, $4.2\text{E}+6$, $2.1\text{E}+9$ in single precision and $4.7\text{E}+7$, $2.3\text{E}+15$ and $2.1\text{E}+9$ in double precision. This makes $U2$ limiting in single precision and $U3$ limiting in double precision. This means that one can expect to retain, in the worst cases on IEEE machines, no digits in single precision and only 6 digits in double precision. Similar considerations hold for other machines.

The approximate relative error in the magnitude of a complex Bessel function can be expressed as $P \cdot 10^{**S}$ where $P = \text{MAX}(\text{UNIT ROUNDOFF}, 1.0\text{E}-18)$ is the nominal precision and 10^{**S} represents the increase in error due to argument reduction in the elementary functions. Here, $S = \text{MAX}(1, \text{ABS}(\text{LOG10}(\text{ABS}(Z))), \text{ABS}(\text{LOG10}(\text{FNU})))$ approximately (i.e., $S = \text{MAX}(1, \text{ABS}(\text{EXPONENT OF ABS}(Z), \text{ABS}(\text{EXPONENT OF FNU})))$). However, the phase angle may have only absolute accuracy. This is most likely to occur when one component (in magnitude) is larger than the other by several orders of magnitude. If one component is 10^{**K} larger than the other, then one can expect only $\text{MAX}(\text{ABS}(\text{LOG10}(P)) - K, 0)$ significant digits; or, stated another way, when K exceeds the exponent of P , no significant digits remain in the smaller component. However, the phase angle retains absolute accuracy because, in complex arithmetic with precision P , the smaller component will not (as a rule) decrease below P times the

magnitude of the larger component. In these extreme cases, the principal phase angle is on the order of $+P$, $-P$, $\pi/2-P$, or $-\pi/2+P$.

- ***REFERENCES
1. M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions, National Bureau of Standards Applied Mathematics Series 55, U. S. Department of Commerce, Tenth Printing (1972) or later.
 2. D. E. Amos, Computation of Bessel Functions of Complex Argument, Report SAND83-0086, Sandia National Laboratories, Albuquerque, NM, May 1983.
 3. D. E. Amos, Computation of Bessel Functions of Complex Argument and Large Order, Report SAND83-0643, Sandia National Laboratories, Albuquerque, NM, May 1983.
 4. D. E. Amos, A Subroutine Package for Bessel Functions of a Complex Argument and Nonnegative Order, Report SAND85-1018, Sandia National Laboratory, Albuquerque, NM, May 1985.
 5. D. E. Amos, A portable package for Bessel functions of a complex argument and nonnegative order, ACM Transactions on Mathematical Software, 12 (September 1986), pp. 265-273.

***ROUTINES CALLED CBINU, ILMACH, RLMACH

***REVISION HISTORY (YMMDD)

830501 DATE WRITTEN

890801 REVISION DATE from Version 3.2

910415 Prologue converted to Version 4.0 format. (BAB)

920128 Category corrected. (WRB)

920811 Prologue revised. (DWL)

END PROLOGUE

CBESK

```

SUBROUTINE CBESK (Z, FNU, KODE, N, CY, NZ, IERR)
***BEGIN PROLOGUE  CBESK
***PURPOSE  Compute a sequence of the Bessel functions  $K(a,z)$  for
             complex argument  $z$  and real nonnegative orders  $a=b, b+1,$ 
              $b+2, \dots$  where  $b>0$ . A scaling option is available to
             help avoid overflow.
***LIBRARY    SLATEC
***CATEGORY   C10B4
***TYPE       COMPLEX (CBESK-C, ZBESK-C)
***KEYWORDS   BESSEL FUNCTIONS OF COMPLEX ARGUMENT, K BESSEL FUNCTIONS,
             MODIFIED BESSEL FUNCTIONS
***AUTHOR     Amos, D. E., (SNL)
***DESCRIPTION

```

On $KODE=1$, CBESK computes an N member sequence of complex Bessel functions $CY(L)=K(FNU+L-1,Z)$ for real nonnegative orders $FNU+L-1$, $L=1, \dots, N$ and complex $Z \neq 0$ in the cut plane $-\pi < \arg(Z) \leq \pi$. On $KODE=2$, CBESJ returns the scaled functions

$$CY(L) = \exp(Z) * K(FNU+L-1, Z), \quad L=1, \dots, N$$

which remove the exponential growth in both the left and right half planes as Z goes to infinity. Definitions and notation are found in the NBS Handbook of Mathematical Functions (Ref. 1).

Input

Z - Nonzero argument of type COMPLEX
 FNU - Initial order of type REAL, $FNU \geq 0$
 $KODE$ - A parameter to indicate the scaling option
 $KODE=1$ returns
 $CY(L)=K(FNU+L-1,Z)$, $L=1, \dots, N$
 $=2$ returns
 $CY(L)=K(FNU+L-1,Z)*EXP(Z)$, $L=1, \dots, N$
 N - Number of terms in the sequence, $N \geq 1$

Output

CY - Result vector of type COMPLEX
 NZ - Number of underflows set to zero
 $NZ=0$ Normal return
 $NZ>0$ $CY(L)=0$ for NZ values of L (if $\text{Re}(Z)>0$ then $CY(L)=0$ for $L=1, \dots, NZ$; in the complementary half plane the underflows may not be in an uninterrupted sequence)
 $IERR$ - Error flag
 $IERR=0$ Normal return - COMPUTATION COMPLETED
 $IERR=1$ Input error - NO COMPUTATION
 $IERR=2$ Overflow - NO COMPUTATION
 ($\text{abs}(Z)$ too small and/or $FNU+N-1$ too large)
 $IERR=3$ Precision warning - COMPUTATION COMPLETED
 (Result has half precision or less because $\text{abs}(Z)$ or $FNU+N-1$ is large)
 $IERR=4$ Precision error - NO COMPUTATION
 (Result has no precision because

abs(Z) or FNU+N-1 is too large)
 IERR=5 Algorithmic error - NO COMPUTATION
 (Termination condition not met)

*Long Description:

Equations of the reference are implemented to compute $K(a, z)$ for small orders a and $a+1$ in the right half plane $\text{Re}(z) \geq 0$. Forward recurrence generates higher orders. The formula

$$K(a, z \exp(it)) = \exp(-t) K(a, z) - t I(a, z), \quad \text{Re}(z) > 0$$

$$t = i\pi \text{ or } -i\pi$$

continues K to the left half plane.

For large orders, $K(a, z)$ is computed by means of its uniform asymptotic expansion.

For negative orders, the formula

$$K(-a, z) = K(a, z)$$

can be used.

CBESK assumes that a significant digit sinh function is available.

In most complex variable computation, one must evaluate elementary functions. When the magnitude of Z or $\text{FNU}+N-1$ is large, losses of significance by argument reduction occur. Consequently, if either one exceeds $U1 = \text{SQRT}(0.5/\text{UR})$, then losses exceeding half precision are likely and an error flag $\text{IERR}=3$ is triggered where $\text{UR} = \text{R1MACH}(4) = \text{UNIT ROUND OFF}$. Also, if either is larger than $U2 = 0.5/\text{UR}$, then all significance is lost and $\text{IERR}=4$. In order to use the INT function, arguments must be further restricted not to exceed the largest machine integer, $U3 = \text{I1MACH}(9)$. Thus, the magnitude of Z and $\text{FNU}+N-1$ is restricted by $\text{MIN}(U2, U3)$. In IEEE arithmetic, $U1, U2$, and $U3$ approximate $2.0\text{E}+3$, $4.2\text{E}+6$, $2.1\text{E}+9$ in single precision and $4.7\text{E}+7$, $2.3\text{E}+15$ and $2.1\text{E}+9$ in double precision. This makes $U2$ limiting in single precision and $U3$ limiting in double precision. This means that one can expect to retain, in the worst cases on IEEE machines, no digits in single precision and only 6 digits in double precision. Similar considerations hold for other machines.

The approximate relative error in the magnitude of a complex Bessel function can be expressed as $P \cdot 10^{**S}$ where $P = \text{MAX}(\text{UNIT ROUND OFF}, 1.0\text{E}-18)$ is the nominal precision and 10^{**S} represents the increase in error due to argument reduction in the elementary functions. Here, $S = \text{MAX}(1, \text{ABS}(\text{LOG}_{10}(\text{ABS}(Z))), \text{ABS}(\text{LOG}_{10}(\text{FNU})))$ approximately (i.e., $S = \text{MAX}(1, \text{ABS}(\text{EXPONENT OF } \text{ABS}(Z)), \text{ABS}(\text{EXPONENT OF } \text{FNU}))$). However, the phase angle may have only absolute accuracy. This is most likely to occur when one component (in magnitude) is larger than the other by several orders of magnitude. If one component is 10^{**K} larger than the other, then one can expect only $\text{MAX}(\text{ABS}(\text{LOG}_{10}(P)) - K, 0)$ significant digits; or, stated another way, when K exceeds the exponent of P , no significant digits remain in the smaller component. However, the phase angle retains absolute accuracy

because, in complex arithmetic with precision P, the smaller component will not (as a rule) decrease below P times the magnitude of the larger component. In these extreme cases, the principal phase angle is on the order of +P, -P, $\pi/2-P$, or $-\pi/2+P$.

- ***REFERENCES
1. M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions, National Bureau of Standards Applied Mathematics Series 55, U. S. Department of Commerce, Tenth Printing (1972) or later.
 2. D. E. Amos, Computation of Bessel Functions of Complex Argument, Report SAND83-0086, Sandia National Laboratories, Albuquerque, NM, May 1983.
 3. D. E. Amos, Computation of Bessel Functions of Complex Argument and Large Order, Report SAND83-0643, Sandia National Laboratories, Albuquerque, NM, May 1983.
 4. D. E. Amos, A Subroutine Package for Bessel Functions of a Complex Argument and Nonnegative Order, Report SAND85-1018, Sandia National Laboratory, Albuquerque, NM, May 1985.
 5. D. E. Amos, A portable package for Bessel functions of a complex argument and nonnegative order, ACM Transactions on Mathematical Software, 12 (September 1986), pp. 265-273.

***ROUTINES CALLED CACON, CBKNU, CBUNK, CUOIK, ILMACH, RLMACH

***REVISION HISTORY (YYMMDD)

830501 DATE WRITTEN

890801 REVISION DATE from Version 3.2

910415 Prologue converted to Version 4.0 format. (BAB)

920128 Category corrected. (WRB)

920811 Prologue revised. (DWL)

END PROLOGUE

CBESY

```
SUBROUTINE CBESY (Z, FNU, KODE, N, CY, NZ, CWRK, IERR)
***BEGIN PROLOGUE  CBESY
***PURPOSE  Compute a sequence of the Bessel functions Y(a,z) for
             complex argument z and real nonnegative orders a=b,b+1,
             b+2,... where b>0.  A scaling option is available to
             help avoid overflow.
***LIBRARY    SLATEC
***CATEGORY   C10A4
***TYPE       COMPLEX (CBESY-C, ZBESY-C)
***KEYWORDS   BESSEL FUNCTIONS OF COMPLEX ARGUMENT,
             BESSEL FUNCTIONS OF SECOND KIND, WEBER'S FUNCTION,
             Y BESSEL FUNCTIONS
***AUTHOR     Amos, D. E., (SNL)
***DESCRIPTION
```

On KODE=1, CBESY computes an N member sequence of complex Bessel functions $CY(L)=Y(FNU+L-1,Z)$ for real nonnegative orders $FNU+L-1$, $L=1,\dots,N$ and complex Z in the cut plane $-\pi < \arg(Z) \leq \pi$. On KODE=2, CBESY returns the scaled functions

$$CY(L) = \exp(-\text{abs}(Y)) * Y(FNU+L-1, Z), \quad L=1,\dots,N, \quad Y=\text{Im}(Z)$$

which remove the exponential growth in both the upper and lower half planes as Z goes to infinity. Definitions and notation are found in the NBS Handbook of Mathematical Functions (Ref. 1).

Input

Z - Nonzero argument of type COMPLEX
FNU - Initial order of type REAL, $FNU \geq 0$
KODE - A parameter to indicate the scaling option
 KODE=1 returns
 $CY(L)=Y(FNU+L-1,Z)$, $L=1,\dots,N$
 =2 returns
 $CY(L)=Y(FNU+L-1,Z)*\exp(-\text{abs}(Y))$, $L=1,\dots,N$
 where $Y=\text{Im}(Z)$
N - Number of terms in the sequence, $N \geq 1$
CWRK - A work vector of type COMPLEX and dimension N

Output

CY - Result vector of type COMPLEX
NZ - Number of underflows set to zero
 NZ=0 Normal return
 NZ>0 $CY(L)=0$ for NZ values of L, usually on
 KODE=2 (the underflows may not be in an
 uninterrupted sequence)
IERR - Error flag
 IERR=0 Normal return - COMPUTATION COMPLETED
 IERR=1 Input error - NO COMPUTATION
 IERR=2 Overflow - NO COMPUTATION
 (abs(Z) too small and/or $FNU+N-1$
 too large)
 IERR=3 Precision warning - COMPUTATION COMPLETED
 (Result has half precision or less
 because abs(Z) or $FNU+N-1$ is large)

```

IERR=4 Precision error - NO COMPUTATION
      (Result has no precision because
      abs(Z) or FNU+N-1 is too large)
IERR=5 Algorithmic error - NO COMPUTATION
      (Termination condition not met)

```

***Long Description:**

The computation is carried out by the formula

$$Y(a,z) = (H(1,a,z) - H(2,a,z))/(2*i)$$

where the Hankel functions are computed as described in CBESH.

For negative orders, the formula

$$Y(-a,z) = Y(a,z)*\cos(a*\pi) + J(a,z)*\sin(a*\pi)$$

can be used. However, for large orders close to half odd integers the function changes radically. When a is a large positive half odd integer, the magnitude of $Y(-a,z)=J(a,z)*\sin(a*\pi)$ is a large negative power of ten. But when a is not a half odd integer, $Y(a,z)$ dominates in magnitude with a large positive power of ten and the most that the second term can be reduced is by unit roundoff from the coefficient. Thus, wide changes can occur within unit roundoff of a large half odd integer. Here, large means $a > \text{abs}(z)$.

In most complex variable computation, one must evaluate elementary functions. When the magnitude of Z or $FNU+N-1$ is large, losses of significance by argument reduction occur. Consequently, if either one exceeds $U1=\text{SQRT}(0.5/UR)$, then losses exceeding half precision are likely and an error flag $IERR=3$ is triggered where $UR=R1MACH(4)=\text{UNIT ROUNDOFF}$. Also, if either is larger than $U2=0.5/UR$, then all significance is lost and $IERR=4$. In order to use the INT function, arguments must be further restricted not to exceed the largest machine integer, $U3=I1MACH(9)$. Thus, the magnitude of Z and $FNU+N-1$ is restricted by $\text{MIN}(U2,U3)$. In IEEE arithmetic, $U1, U2$, and $U3$ approximate $2.0E+3$, $4.2E+6$, $2.1E+9$ in single precision and $4.7E+7$, $2.3E+15$ and $2.1E+9$ in double precision. This makes $U2$ limiting in single precision and $U3$ limiting in double precision. This means that one can expect to retain, in the worst cases on IEEE machines, no digits in single precision and only 6 digits in double precision. Similar considerations hold for other machines.

The approximate relative error in the magnitude of a complex Bessel function can be expressed as $P*10^{**S}$ where $P=\text{MAX}(\text{UNIT ROUNDOFF}, 1.0E-18)$ is the nominal precision and 10^{**S} represents the increase in error due to argument reduction in the elementary functions. Here, $S=\text{MAX}(1, \text{ABS}(\text{LOG10}(\text{ABS}(Z))), \text{ABS}(\text{LOG10}(FNU)))$ approximately (i.e., $S=\text{MAX}(1, \text{ABS}(\text{EXPONENT OF } \text{ABS}(Z)), \text{ABS}(\text{EXPONENT OF } FNU))$). However, the phase angle may have only absolute accuracy. This is most likely to occur when one component (in magnitude) is larger than the other by several orders of magnitude. If one component is 10^{**K} larger than the other, then one can expect only $\text{MAX}(\text{ABS}(\text{LOG10}(P))-K, 0)$ significant digits; or, stated another way, when K exceeds the exponent of P , no significant digits remain in the smaller

component. However, the phase angle retains absolute accuracy because, in complex arithmetic with precision P , the smaller component will not (as a rule) decrease below P times the magnitude of the larger component. In these extreme cases, the principal phase angle is on the order of $+P$, $-P$, $\pi/2-P$, or $-\pi/2+P$.

- ***REFERENCES
1. M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions, National Bureau of Standards Applied Mathematics Series 55, U. S. Department of Commerce, Tenth Printing (1972) or later.
 2. D. E. Amos, Computation of Bessel Functions of Complex Argument, Report SAND83-0086, Sandia National Laboratories, Albuquerque, NM, May 1983.
 3. D. E. Amos, Computation of Bessel Functions of Complex Argument and Large Order, Report SAND83-0643, Sandia National Laboratories, Albuquerque, NM, May 1983.
 4. D. E. Amos, A Subroutine Package for Bessel Functions of a Complex Argument and Nonnegative Order, Report SAND85-1018, Sandia National Laboratory, Albuquerque, NM, May 1985.
 5. D. E. Amos, A portable package for Bessel functions of a complex argument and nonnegative order, ACM Transactions on Mathematical Software, 12 (September 1986), pp. 265-273.

***ROUTINES CALLED CBESH, ILMACH, RLMACH

***REVISION HISTORY (YYMMDD)

830501 DATE WRITTEN

890801 REVISION DATE from Version 3.2

910415 Prologue converted to Version 4.0 format. (BAB)

920128 Category corrected. (WRB)

920811 Prologue revised. (DWL)

END PROLOGUE

CBETA

```
      COMPLEX FUNCTION CBETA (A, B)
***BEGIN PROLOGUE  CBETA
***PURPOSE  Compute the complete Beta function.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C7B
***TYPE      COMPLEX (BETA-S, DBETA-D, CBETA-C)
***KEYWORDS  COMPLETE BETA FUNCTION, FNLIB, SPECIAL FUNCTIONS
***AUTHOR   Fullerton, W., (LANL)
***DESCRIPTION

      CBETA computes the complete beta function of complex parameters A
      and B.
      Input Parameters:
            A   complex and the real part of A positive
            B   complex and the real part of B positive

***REFERENCES  (NONE)
***ROUTINES CALLED  CGAMMA, CLBETA, GAMLIM, XERMSG
***REVISION HISTORY  (YMMDD)
      770701  DATE WRITTEN
      890206  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
      900326  Removed duplicate information from DESCRIPTION section.
              (WRB)
      900727  Added EXTERNAL statement.  (WRB)
      END PROLOGUE
```

CBIRY

```

SUBROUTINE CBIRY (Z, ID, KODE, BI, IERR)
***BEGIN PROLOGUE  CBIRY
***PURPOSE  Compute the Airy function Bi(z) or its derivative dBi/dz
             for complex argument z.  A scaling option is available
             to help avoid overflow.
***LIBRARY    SLATEC
***CATEGORY   C10D
***TYPE       COMPLEX (CBIRY-C, ZBIRY-C)
***KEYWORDS   AIRY FUNCTION, BESSEL FUNCTION OF ORDER ONE THIRD,
             BESSEL FUNCTION OF ORDER TWO THIRDS
***AUTHOR    Amos, D. E., (SNL)
***DESCRIPTION

```

On KODE=1, CBIRY computes the complex Airy function Bi(z) or its derivative dBi/dz on ID=0 or ID=1 respectively. On KODE=2, a scaling option $\exp(\text{abs}(\text{Re}(\text{zeta}))) \cdot \text{Bi}(z)$ or $\exp(\text{abs}(\text{Re}(\text{zeta}))) \cdot \text{dBi}/\text{dz}$ is provided to remove the exponential behavior in both the left and right half planes where $\text{zeta} = (2/3) \cdot z^{2/3}$.

The Airy functions Bi(z) and dBi/dz are analytic in the whole z-plane, and the scaling option does not destroy this property.

Input

```

Z          - Argument of type COMPLEX
ID         - Order of derivative, ID=0 or ID=1
KODE       - A parameter to indicate the scaling option
              KODE=1  returns
                  BI=Bi(z)   on ID=0
                  BI=dBi/dz  on ID=1
                  at z=Z
              =2  returns
                  BI=exp(abs(Re(zeta)))*Bi(z)   on ID=0
                  BI=exp(abs(Re(zeta)))*dBi/dz  on ID=1
                  at z=Z where zeta=(2/3)*z**(3/2)

```

Output

```

BI         - Result of type COMPLEX
IERR       - Error flag
              IERR=0  Normal return      - COMPUTATION COMPLETED
              IERR=1  Input error        - NO COMPUTATION
              IERR=2  Overflow            - NO COMPUTATION
                  (Re(Z) too large with KODE=1)
              IERR=3  Precision warning - COMPUTATION COMPLETED
                  (Result has less than half precision)
              IERR=4  Precision error    - NO COMPUTATION
                  (Result has no precision)
              IERR=5  Algorithmic error - NO COMPUTATION
                  (Termination condition not met)

```

*Long Description:

Bi(z) and dBi/dz are computed from I Bessel functions by

$$\text{Bi}(z) = c \cdot \sqrt{z} \cdot (I(-1/3, \text{zeta}) + I(1/3, \text{zeta}))$$

```

dBi/dz = c*    z    *( I(-2/3,zeta) + I(2/3,zeta) )
      c =  1/sqrt(3)
      zeta = (2/3)*z**(3/2)

```

when $\text{abs}(z) > 1$ and from power series when $\text{abs}(z) \leq 1$.

In most complex variable computation, one must evaluate elementary functions. When the magnitude of Z is large, losses of significance by argument reduction occur. Consequently, if the magnitude of $ZETA = (2/3) * Z^{3/2}$ exceeds $U1 = \text{SQRT}(0.5/UR)$, then losses exceeding half precision are likely and an error flag $IERR=3$ is triggered where $UR=R1MACH(4)=\text{UNIT ROUND OFF}$. Also, if the magnitude of ZETA is larger than $U2 = 0.5/UR$, then all significance is lost and $IERR=4$. In order to use the INT function, ZETA must be further restricted not to exceed $U3 = 1/MACH(9) = \text{LARGEST INTEGER}$. Thus, the magnitude of ZETA must be restricted by $\text{MIN}(U2, U3)$. In IEEE arithmetic, $U1, U2$, and $U3$ are approximately $2.0E+3$, $4.2E+6$, $2.1E+9$ in single precision and $4.7E+7$, $2.3E+15$, $2.1E+9$ in double precision. This makes $U2$ limiting in single precision and $U3$ limiting in double precision. This means that the magnitude of Z cannot exceed approximately $3.4E+4$ in single precision and $2.1E+6$ in double precision. This also means that one can expect to retain, in the worst cases on 32-bit machines, no digits in single precision and only 6 digits in double precision.

The approximate relative error in the magnitude of a complex Bessel function can be expressed as $P * 10^{**S}$ where $P = \text{MAX}(\text{UNIT ROUND OFF}, 1.0E-18)$ is the nominal precision and 10^{**S} represents the increase in error due to argument reduction in the elementary functions. Here, $S = \text{MAX}(1, \text{ABS}(\text{LOG}_{10}(\text{ABS}(Z))), \text{ABS}(\text{LOG}_{10}(\text{FNU})))$ approximately (i.e., $S = \text{MAX}(1, \text{ABS}(\text{EXPONENT OF ABS}(Z), \text{ABS}(\text{EXPONENT OF FNU})))$). However, the phase angle may have only absolute accuracy. This is most likely to occur when one component (in magnitude) is larger than the other by several orders of magnitude. If one component is 10^{**K} larger than the other, then one can expect only $\text{MAX}(\text{ABS}(\text{LOG}_{10}(P)) - K, 0)$ significant digits; or, stated another way, when K exceeds the exponent of P, no significant digits remain in the smaller component. However, the phase angle retains absolute accuracy because, in complex arithmetic with precision P, the smaller component will not (as a rule) decrease below P times the magnitude of the larger component. In these extreme cases, the principal phase angle is on the order of $+P$, $-P$, $\text{PI}/2 - P$, or $-\text{PI}/2 + P$.

- ***REFERENCES
1. M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions, National Bureau of Standards Applied Mathematics Series 55, U. S. Department of Commerce, Tenth Printing (1972) or later.
 2. D. E. Amos, Computation of Bessel Functions of Complex Argument and Large Order, Report SAND83-0643, Sandia National Laboratories, Albuquerque, NM, May 1983.
 3. D. E. Amos, A Subroutine Package for Bessel Functions of a Complex Argument and Nonnegative Order, Report SAND85-1018, Sandia National Laboratory, Albuquerque, NM, May 1985.
 4. D. E. Amos, A portable package for Bessel functions SLATEC2 (AAAAAA through D9UPAK) - 150

of a complex argument and nonnegative order, ACM
Transactions on Mathematical Software, 12 (September
1986), pp. 265-273.

***ROUTINES CALLED CBINU, ILMACH, RLMACH

***REVISION HISTORY (YMMDD)

830501 DATE WRITTEN

890801 REVISION DATE from Version 3.2

910415 Prologue converted to Version 4.0 format. (BAB)

920128 Category corrected. (WRB)

920811 Prologue revised. (DWL)

END PROLOGUE

CBLKTR

```

      SUBROUTINE CBLKTR (IFLG, NP, N, AN, BN, CN, MP, M, AM, BM, CM,
+      IDIMY, Y, IERROR, W)
***BEGIN PROLOGUE  CBLKTR
***PURPOSE  Solve a block tridiagonal system of linear equations
              (usually resulting from the discretization of separable
              two-dimensional elliptic equations).
***LIBRARY    SLATEC (FISHPACK)
***CATEGORY   I2B4B
***TYPE       COMPLEX (BLKTRI-S, CBLKTR-C)
***KEYWORDS   ELLIPTIC PDE, FISHPACK, TRIDIAGONAL LINEAR SYSTEM
***AUTHOR     Adams, J., (NCAR)
              Swarztrauber, P. N., (NCAR)
              Sweet, R., (NCAR)
***DESCRIPTION

```

Subroutine CBLKTR is a complex version of subroutine BLKTRI.
Both subroutines solve a system of linear equations of the form

$$\begin{aligned}
 &AN(J)*X(I,J-1) + AM(I)*X(I-1,J) + (BN(J)+BM(I))*X(I,J) \\
 &+ CN(J)*X(I,J+1) + CM(I)*X(I+1,J) = Y(I,J)
 \end{aligned}$$

For $I = 1, 2, \dots, M$ and $J = 1, 2, \dots, N$.

$I+1$ and $I-1$ are evaluated modulo M and $J+1$ and $J-1$ modulo N , i.e.,

$$\begin{aligned}
 X(I,0) &= X(I,N), & X(I,N+1) &= X(I,1), \\
 X(0,J) &= X(M,J), & X(M+1,J) &= X(1,J).
 \end{aligned}$$

These equations usually result from the discretization of separable elliptic equations. Boundary conditions may be Dirichlet, Neumann, or periodic.

* * * * * On INPUT * * * * *

IFLG

- = 0 Initialization only. Certain quantities that depend on NP, N, AN, BN, and CN are computed and stored in the work array W.
- = 1 The quantities that were computed in the initialization are used to obtain the solution $X(I,J)$.

NOTE A call with IFLG=0 takes approximately one half the time time as a call with IFLG = 1. However, the initialization does not have to be repeated unless NP, N, AN, BN, or CN change.

NP

- = 0 If AN(1) and CN(N) are not zero, which corresponds to periodic boundary conditions.
- = 1 If AN(1) and CN(N) are zero.

N

The number of unknowns in the J-direction. N must be greater than 4. The operation count is proportional to $MN \log_2(N)$, hence

N should be selected less than or equal to M.

AN,BN,CN

Real one-dimensional arrays of length N that specify the coefficients in the linear equations given above.

MP

= 0 If AM(1) and CM(M) are not zero, which corresponds to periodic boundary conditions.
= 1 If AM(1) = CM(M) = 0 .

M

The number of unknowns in the I-direction. M must be greater than 4.

AM,BM,CM

Complex one-dimensional arrays of length M that specify the coefficients in the linear equations given above.

IDIMY

The row (or first) dimension of the two-dimensional array Y as it appears in the program calling BLKTTRI. This parameter is used to specify the variable dimension of Y. IDIMY must be at least M.

Y

A complex two-dimensional array that specifies the values of the right side of the linear system of equations given above. Y must be dimensioned Y(IDIMY,N) with IDIMY .GE. M.

W

A one-dimensional array that must be provided by the user for work space.

If NP=1 define $K = \text{INT}(\log_2(N)) + 1$ and set $L = 2^{**}(K+1)$ then W must have dimension $(K-2)*L + K + 5 + \text{MAX}(2N, 12M)$

If NP=0 define $K = \text{INT}(\log_2(N-1)) + 1$ and set $L = 2^{**}(K+1)$ then W must have dimension $(K-2)*L + K + 5 + 2N + \text{MAX}(2N, 12M)$

****IMPORTANT**** For purposes of checking, the required dimension of W is computed by BLKTTRI and stored in W(1) in floating point format.

* * * * * On Output * * * * *

Y

Contains the solution X.

IERROR

An error flag that indicates invalid input parameters. Except for number zero, a solution is not attempted.

= 0 No error.
= 1 M is less than 5.
= 2 N is less than 5.
= 3 IDIMY is less than M.
= 4 BLKTTRI failed while computing results that depend on the coefficient arrays AN, BN, CN. Check these arrays.
= 5 AN(J)*CN(J-1) is less than 0 for some J. Possible reasons for this condition are

1. The arrays AN and CN are not correct.
2. Too large a grid spacing was used in the discretization of the elliptic equation.
3. The linear equations resulted from a partial differential equation which was not elliptic.

W

Contains intermediate values that must not be destroyed if CBLKTR will be called again with IFLG=1. W(1) contains the number of locations required by W in floating point format.

*Long Description:

* * * * *	Program Specifications	* * * * *
Dimension of Arguments	AN(N),BN(N),CN(N),AM(M),BM(M),CM(M),Y(IDIMY,N) W(see argument list)	
Latest Revision	June 1979	
Required Subprograms	CBLKTR,CBLKT1,PROC,PROCP,CPROC,CPROCP,CCMPB,INXCA, INXCB,INXCC,CPADD,PGSF,PPGSF,PPPSF,BCRH,TEVLC, R1MACH	
Special Conditions	The algorithm may fail if $ABS(BM(I)+BN(J))$ is less than $ABS(AM(I))+ABS(AN(J))+ABS(CM(I))+ABS(CN(J))$ for some I and J. The algorithm will also fail if $AN(J)*CN(J-1)$ is less than zero for some J. See the description of the output parameter IERROR.	
Common Blocks	CCBLK	
I/O	NONE	
Precision	Single	
Specialist	Paul Swarztrauber	
Language	FORTRAN	
History	CBLKTR is a complex version of BLKTRI (version 3)	
Algorithm	Generalized Cyclic Reduction (see reference below)	
Space Required	CONTROL DATA 7600	
Portability	American National Standards Institute FORTRAN. The machine accuracy is set using function R1MACH.	
Required Resident Routines	NONE	
References	Swarztrauber,P. and R. SWEET, 'Efficient Fortran Subprograms for the solution of elliptic equations' NCAR TN/IA-109, July, 1975, 138 PP.	

SWARZTRAUBER P. , 'A Direct Method for The Discrete
Solution of Separable Elliptic Equations', SIAM
J. Numer. Anal., 11(1974) PP. 1136-1150.

* * * * *

***REFERENCES P. N. Swarztrauber and R. Sweet, Efficient Fortran
subprograms for the solution of elliptic equations,
NCAR TN/IA-109, July 1975, 138 pp.
P. N. Swarztrauber, A direct method for the discrete
solution of separable elliptic equations, SIAM Journal
on Numerical Analysis 11, (1974), pp. 1136-1150.

***ROUTINES CALLED CBLKT1, CCMPB, CPROC, CPROCP, PROC, PROCP

***COMMON BLOCKS CCBLK

***REVISION HISTORY (YYMMDD)

801001 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890531 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CBRT

```
      FUNCTION CBRT (X)
***BEGIN PROLOGUE  CBRT
***PURPOSE  Compute the cube root.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C2
***TYPE      SINGLE PRECISION (CBRT-S, DCBRT-D, CCBRT-C)
***KEYWORDS  CUBE ROOT, ELEMENTARY FUNCTIONS, FNLIB, ROOTS
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION

      CBRT(X) calculates the cube root of X.

***REFERENCES  (NONE)
***ROUTINES CALLED  R1MACH, R9PAK, R9UPAK
***REVISION HISTORY  (YYMMDD)
      770601  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890531  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      END PROLOGUE
```

CCBRT

```
      COMPLEX FUNCTION CCBRT (Z)
***BEGIN PROLOGUE  CCBRT
***PURPOSE  Compute the cube root.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C2
***TYPE      COMPLEX (CBRT-S, DCBRT-D, CCBRT-C)
***KEYWORDS  CUBE ROOT, ELEMENTARY FUNCTIONS, FNLIB, ROOTS
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION
```

CCBRT(Z) calculates the complex cube root of Z. The principal root for which $-\pi .LT. \arg(Z) .LE. +\pi$ is returned.

```
***REFERENCES  (NONE)
***ROUTINES CALLED  CARG, CBRT
***REVISION HISTORY  (YYMMDD)
      770401  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890531  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      END PROLOGUE
```

CCHDC

```
SUBROUTINE CCHDC (A, LDA, P, WORK, JPVT, JOB, INFO)
***BEGIN PROLOGUE  CCHDC
***PURPOSE  Compute the Cholesky decomposition of a positive definite
             matrix.  A pivoting option allows the user to estimate the
             condition number of a positive definite matrix or determine
             the rank of a positive semidefinite matrix.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2D1B
***TYPE      COMPLEX (SCHDC-S, DCHDC-D, CCHDC-C)
***KEYWORDS  CHOLESKY DECOMPOSITION, LINEAR ALGEBRA, LINPACK, MATRIX,
             POSITIVE DEFINITE
***AUTHOR    Dongarra, J., (ANL)
             Stewart, G. W., (U. of Maryland)
***DESCRIPTION
```

CCHDC computes the Cholesky decomposition of a positive definite matrix. A pivoting option allows the user to estimate the condition of a positive definite matrix or determine the rank of a positive semidefinite matrix.

On Entry

A COMPLEX(LDA,P).
A contains the matrix whose decomposition is to be computed. Only the upper half of A need be stored. The lower part of The array A is not referenced.

LDA INTEGER.
LDA is the leading dimension of the array A.

P INTEGER.
P is the order of the matrix.

WORK COMPLEX.
WORK is a work array.

JPVT INTEGER(P).
JPVT contains integers that control the selection of the pivot elements, if pivoting has been requested. Each diagonal element A(K,K) is placed in one of three classes according to the value of JPVT(K)).

If JPVT(K)) .GT. 0, then X(K) is an initial element.

If JPVT(K)) .EQ. 0, then X(K) is a free element.

If JPVT(K)) .LT. 0, then X(K) is a final element.

Before the decomposition is computed, initial elements are moved by symmetric row and column interchanges to the beginning of the array A and final elements to the end. Both initial and final elements are frozen in place during the computation and only free elements are moved. At the K-th stage of the

reduction, if A(K,K) is occupied by a free element it is interchanged with the largest free element A(L,L) with L .GE. K. JPVT is not referenced if JOB .EQ. 0.

JOB INTEGER.
JOB is an integer that initiates column pivoting.
IF JOB .EQ. 0, no pivoting is done.
IF JOB .NE. 0, pivoting is done.

On Return

A A contains in its upper half the Cholesky factor of the matrix A as it has been permuted by pivoting.

JPVT JPVT(J) contains the index of the diagonal element of A that was moved into the J-th position, provided pivoting was requested.

INFO contains the index of the last positive diagonal element of the Cholesky factor.

For positive definite matrices INFO = P is the normal return. For pivoting with positive semidefinite matrices INFO will in general be less than P. However, INFO may be greater than the rank of A, since rounding error can cause an otherwise zero element to be positive. Indefinite systems will always cause INFO to be less than P.

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CAXPY, CSWAP

***REVISION HISTORY (YYMMDD)

790319 DATE WRITTEN

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900326 Removed duplicate information from DESCRIPTION section. (WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CCHDD

```

SUBROUTINE CCHDD (R, LDR, P, X, Z, LDZ, NZ, Y, RHO, C, S, INFO)
***BEGIN PROLOGUE  CCHDD
***PURPOSE  Downdate an augmented Cholesky decomposition or the
            triangular factor of an augmented QR decomposition.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D7B
***TYPE      COMPLEX (SCHDD-S, DCHDD-D, CCHDD-C)
***KEYWORDS  CHOLESKY DECOMPOSITION, DOWNDATED, LINEAR ALGEBRA, LINPACK,
            MATRIX
***AUTHOR  Stewart, G. W., (U. of Maryland)
***DESCRIPTION

```

CCHDD downdates an augmented Cholesky decomposition or the triangular factor of an augmented QR decomposition. Specifically, given an upper triangular matrix R of order P, a row vector X, a column vector Z, and a scalar Y, CCHDD determines a unitary matrix U and a scalar ZETA such that

$$U * \begin{pmatrix} R & Z \\ 0 & ZETA \end{pmatrix} = \begin{pmatrix} RR & ZZ \\ X & Y \end{pmatrix},$$

where RR is upper triangular. If R and Z have been obtained from the factorization of a least squares problem, then RR and ZZ are the factors corresponding to the problem with the observation (X,Y) removed. In this case, if RHO is the norm of the residual vector, then the norm of the residual vector of the downdated problem is $\sqrt{RHO^2 - ZETA^2}$. CCHDD will simultaneously downdate several triplets (Z,Y,RHO) along with R. For a less terse description of what CCHDD does and how it may be applied, see the LINPACK Guide.

The matrix U is determined as the product $U(1)*...*U(P)$ where U(I) is a rotation in the (P+1,I)-plane of the form

$$\begin{pmatrix} C(I) & -CONJG(S(I)) \\ S(I) & C(I) \end{pmatrix}.$$

the rotations are chosen so that C(I) is real.

The user is warned that a given downdating problem may be impossible to accomplish or may produce inaccurate results. For example, this can happen if X is near a vector whose removal will reduce the rank of R. Beware.

On Entry

R COMPLEX(LDR,P), where LDR .GE. P.
R contains the upper triangular matrix that is to be downdated. The part of R below the diagonal is not referenced.

LDR INTEGER.
LDR is the leading dimension of the array R.

p INTEGER.
P is the order of the matrix R.

X COMPLEX(P).
X contains the row vector that is to
be removed from R. X is not altered by CCHDD.

Z COMPLEX(LDZ,NZ), where LDZ .GE. P.
Z is an array of NZ P-vectors which
are to be downdated along with R.

LDZ INTEGER.
LDZ is the leading dimension of the array Z.

NZ INTEGER.
NZ is the number of vectors to be downdated
NZ may be zero, in which case Z, Y, and RHO
are not referenced.

Y COMPLEX(NZ).
Y contains the scalars for the downdating
of the vectors Z. Y is not altered by CCHDD.

RHO REAL(NZ).
RHO contains the norms of the residual
vectors that are to be downdated.

On Return

R
Z
RHO contain the downdated quantities.

C REAL(P).
C contains the cosines of the transforming
rotations.

S COMPLEX(P).
S contains the sines of the transforming
rotations.

INFO INTEGER.
INFO is set as follows.

INFO = 0 if the entire downdating
 was successful.

INFO = -1 if R could not be downdated.
 in this case, all quantities
 are left unaltered.

INFO = 1 if some RHO could not be
 downdated. The offending RHO's are
 set to -1.

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
 Stewart, LINPACK Users' Guide, SIAM, 1979.

SLATEC2 (AAAAAA through D9UPAK) - 161

```
***ROUTINES CALLED  CDOTC, SCNRM2
***REVISION HISTORY  (YYMMDD)
 780814  DATE WRITTEN
 890531  Changed all specific intrinsics to generic.  (WRB)
 890831  Modified array declarations.  (WRB)
 890831  REVISION DATE from Version 3.2
 891214  Prologue converted to Version 4.0 format.  (BAB)
 900326  Removed duplicate information from DESCRIPTION section.
        (WRB)
 920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

CCHEX

```

SUBROUTINE CCHEX (R, LDR, P, K, L, Z, LDZ, NZ, C, S, JOB)
***BEGIN PROLOGUE  CCHEX
***PURPOSE  Update the Cholesky factorization  $A=TRANS(R)*R$  of a
             positive definite matrix A of order P under diagonal
             permutations of the form  $TRANS(E)*A*E$ , where E is a
             permutation matrix.
***LIBRARY    SLATEC (LINPACK)
***CATEGORY  D7B
***TYPE      COMPLEX (SCHEX-S, DCHEX-D, CCHEX-C)
***KEYWORDS  CHOLESKY DECOMPOSITION, EXCHANGE, LINEAR ALGEBRA, LINPACK,
             MATRIX, POSITIVE DEFINITE
***AUTHOR    Stewart, G. W., (U. of Maryland)
***DESCRIPTION

```

CCHEX updates the Cholesky factorization

$$A = CTRANS(R)*R$$

of a positive definite matrix A of order P under diagonal permutations of the form

$$TRANS(E)*A*E$$

where E is a permutation matrix. Specifically, given an upper triangular matrix R and a permutation matrix E (which is specified by K, L, and JOB), CCHEX determines a unitary matrix U such that

$$U*R*E = RR,$$

where RR is upper triangular. At the users option, the transformation U will be multiplied into the array Z. If $A = CTRANS(X)*X$, so that R is the triangular part of the QR factorization of X, then RR is the triangular part of the QR factorization of $X*E$, i.e. X with its columns permuted. For a less terse description of what CCHEX does and how it may be applied, see the LINPACK Guide.

The matrix Q is determined as the product $U(L-K)*...*U(1)$ of plane rotations of the form

$$\begin{pmatrix} C(I) & S(I) \\ 0 & 1 \end{pmatrix} \text{ or } \begin{pmatrix} 1 & 0 \\ -CONJG(S(I)) & C(I) \end{pmatrix},$$

where C(I) is real. The rows these rotations operate on are described below.

There are two types of permutations, which are determined by the value of JOB.

1. Right circular shift (JOB = 1).

The columns are rearranged in the following order.

$$1, \dots, K-1, L, K, K+1, \dots, L-1, L+1, \dots, P.$$

U is the product of L-K rotations $U(I)$, where $U(I)$ acts in the $(L-I, L-I+1)$ -plane.

2. Left circular shift (JOB = 2).

The columns are rearranged in the following order

$1, \dots, K-1, K+1, K+2, \dots, L, K, L+1, \dots, P.$

U is the product of L-K rotations $U(I)$, where $U(I)$ acts in the $(K+I-1, K+I)$ -plane.

On Entry

R COMPLEX(LDR,P), where LDR .GE. P.
R contains the upper triangular factor
that is to be updated. Elements of R
below the diagonal are not referenced.

LDR INTEGER.
LDR is the leading dimension of the array R.

P INTEGER.
P is the order of the matrix R.

K INTEGER.
K is the first column to be permuted.

L INTEGER.
L is the last column to be permuted.
L must be strictly greater than K.

Z COMPLEX(LDZ,NZ), where LDZ .GE. P.
Z is an array of NZ P-vectors into which the
transformation U is multiplied. Z is
not referenced if NZ = 0.

LDZ INTEGER.
LDZ is the leading dimension of the array Z.

NZ INTEGER.
NZ is the number of columns of the matrix Z.

JOB INTEGER.
JOB determines the type of permutation.
 JOB = 1 right circular shift.
 JOB = 2 left circular shift.

On Return

R contains the updated factor.

Z contains the updated matrix Z.

C REAL(P).
C contains the cosines of the transforming rotations.

S COMPLEX(P).
S contains the sines of the transforming rotations.

```
***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
                Stewart, LINPACK Users' Guide, SIAM, 1979.
***ROUTINES CALLED  CROTG
***REVISION HISTORY  (YYMMDD)
  780814  DATE WRITTEN
  890531  Changed all specific intrinsics to generic.  (WRB)
  890831  Modified array declarations.  (WRB)
  890831  REVISION DATE from Version 3.2
  891214  Prologue converted to Version 4.0 format.  (BAB)
  900326  Removed duplicate information from DESCRIPTION section.
          (WRB)
  920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

CCHUD

```

SUBROUTINE CCHUD (R, LDR, P, X, Z, LDZ, NZ, Y, RHO, C, S)
***BEGIN PROLOGUE  CCHUD
***PURPOSE  Update an augmented Cholesky decomposition of the
             triangular part of an augmented QR decomposition.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D7B
***TYPE      COMPLEX (SCHUD-S, DCHUD-D, CCHUD-C)
***KEYWORDS  CHOLESKY DECOMPOSITION, LINEAR ALGEBRA, LINPACK, MATRIX,
             UPDATE
***AUTHOR   Stewart, G. W., (U. of Maryland)
***DESCRIPTION

```

CCHUD updates an augmented Cholesky decomposition of the triangular part of an augmented QR decomposition. Specifically, given an upper triangular matrix R of order P, a row vector X, a column vector Z, and a scalar Y, CCHUD determines a unitary matrix U and a scalar ZETA such that

$$U \begin{pmatrix} R & Z \\ X & Y \end{pmatrix} = \begin{pmatrix} RR & ZZ \\ 0 & ZETA \end{pmatrix},$$

where RR is upper triangular. If R and Z have been obtained from the factorization of a least squares problem, then RR and ZZ are the factors corresponding to the problem with the observation (X,Y) appended. In this case, if RHO is the norm of the residual vector, then the norm of the residual vector of the updated problem is $\text{SQRT}(\text{RHO}^2 + \text{ZETA}^2)$. CCHUD will simultaneously update several triplets (Z,Y,RHO).

For a less terse description of what CCHUD does and how it may be applied see the LINPACK Guide.

The matrix U is determined as the product $U(P)*\dots*U(1)$, where U(I) is a rotation in the (I,P+1) plane of the form

$$\begin{pmatrix} (CI) & S(I) \\ (&) \\ (-\text{CONJG}(S(I)) & (CI) \end{pmatrix}.$$

The rotations are chosen so that C(I) is real.

On Entry

```

R      COMPLEX(LDR,P), where LDR .GE. P.
       R contains the upper triangular matrix
       that is to be updated. The part of R
       below the diagonal is not referenced.

LDR    INTEGER.
       LDR is the leading dimension of the array R.

P      INTEGER.

```

P is the order of the matrix R.

X COMPLEX(P).
X contains the row to be added to R. X is
not altered by CCHUD.

Z COMPLEX(LDZ,NZ), where LDZ .GE. P.
Z is an array containing NZ P-vectors to
be updated with R.

LDZ INTEGER.
LDZ is the leading dimension of the array Z.

NZ INTEGER.
NZ is the number of vectors to be updated
NZ may be zero, in which case Z, Y, and RHO
are not referenced.

Y COMPLEX(NZ).
Y contains the scalars for updating the vectors
Z. Y is not altered by CCHUD.

RHO REAL(NZ).
RHO contains the norms of the residual
vectors that are to be updated. If RHO(J)
is negative, it is left unaltered.

On Return

RC
RHO contain the updated quantities.
Z

C REAL(P).
C contains the cosines of the transforming
rotations.

S COMPLEX(P).
S contains the sines of the transforming
rotations.

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CROTG

***REVISION HISTORY (YYMMDD)

780814 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900326 Removed duplicate information from DESCRIPTION section.
(WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CCOPY

```
SUBROUTINE CCOPY (N, CX, INCX, CY, INCY)
***BEGIN PROLOGUE  CCOPY
***PURPOSE  Copy a vector.
***LIBRARY   SLATEC (BLAS)
***CATEGORY  D1A5
***TYPE      COMPLEX (SCOPY-S, DCOPY-D, CCOPY-C, ICOPY-I)
***KEYWORDS  BLAS, COPY, LINEAR ALGEBRA, VECTOR
***AUTHOR    Lawson, C. L., (JPL)
              Hanson, R. J., (SNLA)
              Kincaid, D. R., (U. of Texas)
              Krogh, F. T., (JPL)
***DESCRIPTION

              B L A S  Subprogram
Description of Parameters

--Input--
  N  number of elements in input vector(s)
  CX  complex vector with N elements
  INCX  storage spacing between elements of CX
  CY  complex vector with N elements
  INCY  storage spacing between elements of CY

--Output--
  CY  copy of vector CX (unchanged if N .LE. 0)

Copy complex CX to complex CY.
For I = 0 to N-1, copy CX(LX+I*INCX) to CY(LY+I*INCY),
where LX = 1 if INCX .GE. 0, else LX = 1+(1-N)*INCX, and LY is
defined in a similar way using INCY.

***REFERENCES  C. L. Lawson, R. J. Hanson, D. R. Kincaid and F. T.
              Krogh, Basic linear algebra subprograms for Fortran
              usage, Algorithm No. 539, Transactions on Mathematical
              Software 5, 3 (September 1979), pp. 308-323.
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
  791001  DATE WRITTEN
  890831  Modified array declarations.  (WRB)
  890831  REVISION DATE from Version 3.2
  891214  Prologue converted to Version 4.0 format.  (BAB)
  920310  Corrected definition of LX in DESCRIPTION.  (WRB)
  920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

CCOSH

```
      COMPLEX FUNCTION CCOSH (Z)
***BEGIN PROLOGUE  CCOSH
***PURPOSE  Compute the complex hyperbolic cosine.
***LIBRARY  SLATEC (FNLIB)
***CATEGORY  C4C
***TYPE      COMPLEX (CCOSH-C)
***KEYWORDS  ELEMENTARY FUNCTIONS, FNLIB, HYPERBOLIC COSINE
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION

      CCOSH(Z) calculates the complex hyperbolic cosine of Z.

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      770401  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890531  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      END PROLOGUE
```

CCOT

```
      COMPLEX FUNCTION CCOT (Z)
***BEGIN PROLOGUE  CCOT
***PURPOSE  Compute the cotangent.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C4A
***TYPE      COMPLEX (COT-S, DCOT-D, CCOT-C)
***KEYWORDS  COTANGENT, ELEMENTARY FUNCTIONS, FNLIB, TRIGONOMETRIC
***AUTHOR    Fullerton, W., (LANL)
***DESCRIPTION
```

CCOT(Z) calculates the complex trigonometric cotangent of Z.

```
***REFERENCES  (NONE)
***ROUTINES CALLED  R1MACH, XERCLR, XERMSG
***REVISION HISTORY  (YYMMDD)
    770401  DATE WRITTEN
    890531  Changed all specific intrinsics to generic.  (WRB)
    890531  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
    900326  Removed duplicate information from DESCRIPTION section.
           (WRB)
END PROLOGUE
```

CDCDOT

```
      COMPLEX FUNCTION CDCDOT (N, CB, CX, INCX, CY, INCY)
***BEGIN PROLOGUE  CDCDOT
***PURPOSE  Compute the inner product of two vectors with extended
            precision accumulation.
***LIBRARY   SLATEC (BLAS)
***CATEGORY  D1A4
***TYPE      COMPLEX (SDSDOT-S, CDCDOT-C)
***KEYWORDS  BLAS, DOT PRODUCT, INNER PRODUCT, LINEAR ALGEBRA, VECTOR
***AUTHOR   Lawson, C. L., (JPL)
            Hanson, R. J., (SNLA)
            Kincaid, D. R., (U. of Texas)
            Krogh, F. T., (JPL)
***DESCRIPTION

            B L A S  Subprogram
Description of Parameters

      --Input--
      N  number of elements in input vector(s)
      CB  complex scalar to be added to inner product
      CX  complex vector with N elements
      INCX  storage spacing between elements of CX
      CY  complex vector with N elements
      INCY  storage spacing between elements of CY

      --Output--
CDCDOT  complex dot product (CB if N .LE. 0)

      Returns complex result with dot product accumulated in D.P.
CDCDOT = CB + sum for I = 0 to N-1 of CX(LX+I*INCY)*CY(LY+I*INCY)
      where LX = 1 if INCX .GE. 0, else LX = 1+(1-N)*INCX, and LY is
      defined in a similar way using INCY.

***REFERENCES  C. L. Lawson, R. J. Hanson, D. R. Kincaid and F. T.
              Krogh, Basic linear algebra subprograms for Fortran
              usage, Algorithm No. 539, Transactions on Mathematical
              Software 5, 3 (September 1979), pp. 308-323.
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      791001  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890531  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      920310  Corrected definition of LX in DESCRIPTION.  (WRB)
      920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

CDOTC

```
      COMPLEX FUNCTION CDOTC (N, CX, INCX, CY, INCY)
***BEGIN PROLOGUE  CDOTC
***PURPOSE  Dot product of two complex vectors using the complex
             conjugate of the first vector.
***LIBRARY   SLATEC (BLAS)
***CATEGORY  D1A4
***TYPE      COMPLEX (CDOTC-C)
***KEYWORDS  BLAS, INNER PRODUCT, LINEAR ALGEBRA, VECTOR
***AUTHOR   Lawson, C. L., (JPL)
             Hanson, R. J., (SNLA)
             Kincaid, D. R., (U. of Texas)
             Krogh, F. T., (JPL)
***DESCRIPTION

             B L A S  Subprogram
Description of Parameters

      --Input--
      N  number of elements in input vector(s)
      CX  complex vector with N elements
      INCX  storage spacing between elements of CX
      CY  complex vector with N elements
      INCY  storage spacing between elements of CY

      --Output--
      CDOTC  complex result (zero if N .LE. 0)

      Returns the dot product of complex CX and CY, using CONJUGATE(CX)
      CDOTC = SUM for I = 0 to N-1 of CONJ(CX(LX+I*INCX))*CY(LY+I*INCY),
      where LX = 1 if INCX .GE. 0, else LX = 1+(1-N)*INCX, and LY is
      defined in a similar way using INCY.

***REFERENCES  C. L. Lawson, R. J. Hanson, D. R. Kincaid and F. T.
               Krogh, Basic linear algebra subprograms for Fortran
               usage, Algorithm No. 539, Transactions on Mathematical
               Software 5, 3 (September 1979), pp. 308-323.
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      791001  DATE WRITTEN
      890831  Modified array declarations.  (WRB)
      890831  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      920310  Corrected definition of LX in DESCRIPTION.  (WRB)
      920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

CDOTU

```
      COMPLEX FUNCTION CDOTU (N, CX, INCX, CY, INCY)
***BEGIN PROLOGUE  CDOTU
***PURPOSE  Compute the inner product of two vectors.
***LIBRARY   SLATEC (BLAS)
***CATEGORY  D1A4
***TYPE      COMPLEX (SDOT-S, DDOT-D, CDOTU-C)
***KEYWORDS  BLAS, INNER PRODUCT, LINEAR ALGEBRA, VECTOR
***AUTHOR   Lawson, C. L., (JPL)
            Hanson, R. J., (SNLA)
            Kincaid, D. R., (U. of Texas)
            Krogh, F. T., (JPL)
***DESCRIPTION

            B L A S  Subprogram
Description of parameters

      --Input--
      N  number of elements in input vector(s)
      CX  complex vector with N elements
      INCX  storage spacing between elements of CX
      CY  complex vector with N elements
      INCY  storage spacing between elements of CY

      --Output--
      CDOTU  complex result (zero if N .LE. 0)

      Returns the dot product of complex CX and CY, no conjugation
      CDOTU = SUM for I = 0 to N-1 of  CX(LX+I*INCX) * CY(LY+I*INCY),
      where LX = 1 if INCX .GE. 0, else LX = 1+(1-N)*INCX, and LY is
      defined in a similar way using INCY.

***REFERENCES  C. L. Lawson, R. J. Hanson, D. R. Kincaid and F. T.
              Krogh, Basic linear algebra subprograms for Fortran
              usage, Algorithm No. 539, Transactions on Mathematical
              Software 5, 3 (September 1979), pp. 308-323.
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      791001  DATE WRITTEN
      890831  Modified array declarations.  (WRB)
      890831  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      920310  Corrected definition of LX in DESCRIPTION.  (WRB)
      920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

CDRIV1

```
      SUBROUTINE CDRIV1 (N, T, Y, F, TOUT, MSTATE, EPS, WORK, LENW,  
8      IERFLG)  
***BEGIN PROLOGUE  CDRIV1  
***PURPOSE  The function of CDRIV1 is to solve N (200 or fewer)  
             ordinary differential equations of the form  
              $dy(I)/dT = F(Y(I),T)$ , given the initial conditions  
              $Y(I) = YI$ .  CDRIV1 allows complex-valued differential  
             equations.  
***LIBRARY    SLATEC (SDRIVE)  
***CATEGORY   I1A2, I1A1B  
***TYPE       COMPLEX (SDRIV1-S, DDRIV1-D, CDRIV1-C)  
***KEYWORDS   COMPLEX VALUED, GEAR'S METHOD, INITIAL VALUE PROBLEMS,  
             ODE, ORDINARY DIFFERENTIAL EQUATIONS, SDRIVE, STIFF  
***AUTHOR     Kahaner, D. K., (NIST)  
             National Institute of Standards and Technology  
             Gaithersburg, MD 20899  
             Sutherland, C. D., (LANL)  
             Mail Stop D466  
             Los Alamos National Laboratory  
             Los Alamos, NM 87545  
***DESCRIPTION
```

Version 92.1

I. CHOOSING THE CORRECT ROUTINE

SDRIV
DDRIV
CDRIV

These are the generic names for three packages for solving initial value problems for ordinary differential equations. SDRIV uses single precision arithmetic. DDRIV uses double precision arithmetic. CDRIV allows complex-valued differential equations, integrated with respect to a single, real, independent variable.

As an aid in selecting the proper program, the following is a discussion of the important options or restrictions associated with each program:

- A. CDRIV1 should be tried first for those routine problems with no more than 200 differential equations (CDRIV2 and CDRIV3 have no such restriction.) Internally this routine has two important technical defaults:
 1. Numerical approximation of the Jacobian matrix of the right hand side is used.
 2. The stiff solver option is used.Most users of CDRIV1 should not have to concern themselves with these details.
- B. CDRIV2 should be considered for those problems for which CDRIV1 is inadequate. For example, CDRIV1 may have difficulty with problems having zero initial conditions and zero derivatives. In this case CDRIV2, with an appropriate value of the parameter EWT, should perform more efficiently. CDRIV2 provides three important additional options:

1. The nonstiff equation solver (as well as the stiff solver) is available.
 2. The root-finding option is available.
 3. The program can dynamically select either the non-stiff or the stiff methods.
- Internally this routine also defaults to the numerical approximation of the Jacobian matrix of the right hand side.

- C. CDRIV3 is the most flexible, and hence the most complex, of the programs. Its important additional features include:
1. The ability to exploit band structure in the Jacobian matrix.
 2. The ability to solve some implicit differential equations, i.e., those having the form:

$$A(Y,T)*dY/dT = F(Y,T).$$
 3. The option of integrating in the one step mode.
 4. The option of allowing the user to provide a routine which computes the analytic Jacobian matrix of the right hand side.
 5. The option of allowing the user to provide a routine which does all the matrix algebra associated with corrections to the solution components.

II. PARAMETERS

The user should use parameter names in the call sequence of CDRIV1 for those quantities whose value may be altered by CDRIV1. The parameters in the call sequence are:

- N = (Input) The number of differential equations, N .LE. 200
- T = (Real) The independent variable. On input for the first call, T is the initial point. On output, T is the point at which the solution is given.
- Y = (Complex) The vector of dependent variables. Y is used as input on the first call, to set the initial values. On output, Y is the computed solution vector. This array Y is passed in the call sequence of the user-provided routine F. Thus parameters required by F can be stored in this array in components N+1 and above. (Note: Changes by the user to the first N components of this array will take effect only after a restart, i.e., after setting MSTATE to +1(-1).)

- F = A subroutine supplied by the user. The name must be declared EXTERNAL in the user's calling program. This subroutine is of the form:

```

      SUBROUTINE F (N, T, Y, YDOT)
      COMPLEX Y(*), YDOT(*)
      .
      .
      YDOT(1) = ...
      .
      .
      YDOT(N) = ...
      END (Sample)

```

This computes $YDOT = F(Y,T)$, the right hand side of the differential equations. Here Y is a vector of length at least N. The actual length of Y is determined by the

user's declaration in the program which calls CDRIV1. Thus the dimensioning of Y in F, while required by FORTRAN convention, does not actually allocate any storage. When this subroutine is called, the first N components of Y are intermediate approximations to the solution components. The user should not alter these values. Here YDOT is a vector of length N. The user should only compute YDOT(I) for I from 1 to N. Normally a return from F passes control back to CDRIV1. However, if the user would like to abort the calculation, i.e., return control to the program which calls CDRIV1, he should set N to zero. CDRIV1 will signal this by returning a value of MSTATE equal to +5(-5). Altering the value of N in F has no effect on the value of N in the call sequence of CDRIV1.

TOUT = (Input, Real) The point at which the solution is desired.

MSTATE = An integer describing the status of integration. The user must initialize MSTATE to +1 or -1. If MSTATE is positive, the routine will integrate past TOUT and interpolate the solution. This is the most efficient mode. If MSTATE is negative, the routine will adjust its internal step to reach TOUT exactly (useful if a singularity exists beyond TOUT.) The meaning of the magnitude of MSTATE:

- 1 (Input) Means the first call to the routine. This value must be set by the user. On all subsequent calls the value of MSTATE should be tested by the user. Unless CDRIV1 is to be reinitialized, only the sign of MSTATE may be changed by the user. (As a convenience to the user who may wish to put out the initial conditions, CDRIV1 can be called with MSTATE=+1(-1), and TOUT=T. In this case the program will return with MSTATE unchanged, i.e., MSTATE=+1(-1).)
- 2 (Output) Means a successful integration. If a normal continuation is desired (i.e., a further integration in the same direction), simply advance TOUT and call again. All other parameters are automatically set.
- 3 (Output)(Unsuccessful) Means the integrator has taken 1000 steps without reaching TOUT. The user can continue the integration by simply calling CDRIV1 again.
- 4 (Output)(Unsuccessful) Means too much accuracy has been requested. EPS has been increased to a value the program estimates is appropriate. The user can continue the integration by simply calling CDRIV1 again.
- 5 (Output)(Unsuccessful) N has been set to zero in SUBROUTINE F.
- 6 (Output)(Successful) For MSTATE negative, T is beyond TOUT. The solution was obtained by interpolation. The user can continue the integration by simply advancing TOUT and calling CDRIV1 again.
- 7 (Output)(Unsuccessful) The solution could not be obtained. The value of IERFLG (see description below) for a "Recoverable" situation indicates the type of difficulty encountered: either an illegal value for a parameter or an inability to continue the solution. For this condition the user should take

corrective action and reset MSTATE to +1(-1) before calling CDRIV1 again. Otherwise the program will terminate the run.

EPS = (Real) On input, the requested relative accuracy in all solution components. On output, the adjusted relative accuracy if the input value was too small. The value of EPS should be set as large as is reasonable, because the amount of work done by CDRIV1 increases as EPS decreases.

WORK
LENW = (Input)
WORK is an array of LENW complex words used internally for temporary storage. The user must allocate space for this array in the calling program by a statement such as

COMPLEX WORK(...)

The length of WORK should be at least $N*N + 11*N + 300$ and LENW should be set to the value used. The contents of WORK should not be disturbed between calls to CDRIV1.

IERFLG = An error flag. The error number associated with a diagnostic message (see Section IV-A below) is the same as the corresponding value of IERFLG. The meaning of IERFLG:

- 0 The routine completed successfully. (No message is issued.)
- 3 (Warning) The number of steps required to reach TOUT exceeds 1000 .
- 4 (Warning) The value of EPS is too small.
- 11 (Warning) For MSTATE negative, T is beyond TOUT. The solution was obtained by interpolation.
- 15 (Warning) The integration step size is below the roundoff level of T. (The program issues this message as a warning but does not return control to the user.)
- 21 (Recoverable) N is greater than 200 .
- 22 (Recoverable) N is not positive.
- 26 (Recoverable) The magnitude of MSTATE is either 0 or greater than 7 .
- 27 (Recoverable) EPS is less than zero.
- 32 (Recoverable) Insufficient storage has been allocated for the WORK array.
- 41 (Recoverable) The integration step size has gone to zero.
- 42 (Recoverable) The integration step size has been reduced about 50 times without advancing the solution. The problem setup may not be correct.
- 999 (Fatal) The magnitude of MSTATE is 7 .

III. USAGE

PROGRAM SAMPLE
EXTERNAL F
COMPLEX ALFA
REAL EPS, T, TOUT

C N is the number of equations
PARAMETER(ALFA = (1.E0, 1.E0), N = 3,
8 LENW = $N*N + 11*N + 300$)
COMPLEX WORK(LENW), Y(N+1)

C Initial point

```

      T = 0.00001E0
C                                     Set initial conditions
      Y(1) = 10.E0
      Y(2) = 0.E0
      Y(3) = 10.E0
C                                     Pass parameter
      Y(4) = ALFA
      TOUT = T
      MSTATE = 1
      EPS = .001E0
10  CALL CDRIV1 (N, T, Y, F, TOUT, MSTATE, EPS, WORK, LENW,
8      IERFLG)
      IF (MSTATE .GT. 2) STOP
      WRITE(*, '(5E12.3)') TOUT, (Y(I), I=1,3)
      TOUT = 10.E0*TOUT
      IF (TOUT .LT. 50.E0) GO TO 10
      END

      SUBROUTINE F (N, T, Y, YDOT)
      COMPLEX ALFA, Y(*), YDOT(*)
      REAL T
      ALFA = Y(N+1)
      YDOT(1) = 1.E0 + ALFA*(Y(2) - Y(1)) - Y(1)*Y(3)
      YDOT(2) = ALFA*(Y(1) - Y(2)) - Y(2)*Y(3)
      YDOT(3) = 1.E0 - Y(3)*(Y(1) + Y(2))
      END

```

IV. OTHER COMMUNICATION TO THE USER

- A. The solver communicates to the user through the parameters above. In addition it writes diagnostic messages through the standard error handling program XERMSG. A complete description of XERMSG is given in "Guide to the SLATEC Common Mathematical Library" by Kirby W. Fong et al.. At installations which do not have this error handling package the short but serviceable routine, XERMSG, available with this package, can be used. That program uses the file named OUTPUT to transmit messages.
- B. The number of evaluations of the right hand side can be found in the WORK array in the location determined by:

$$\text{LENW} - (\text{N} + 50) + 4$$

V. REMARKS

For other information, see Section IV of the writeup for CDRIV3.

```

***REFERENCES  C. W. Gear, Numerical Initial Value Problems in
                Ordinary Differential Equations, Prentice-Hall, 1971.
***ROUTINES CALLED  CDRIV3, XERMSG
***REVISION HISTORY  (YYMMDD)
    790601  DATE WRITTEN
    900329  Initial submission to SLATEC.
    END PROLOGUE

```

CDRIV2

```
      SUBROUTINE CDRIV2 (N, T, Y, F, TOUT, MSTATE, NROOT, EPS, EWT,
      8  MINT, WORK, LENW, IWORK, LENIW, G, IERFLG)
***BEGIN PROLOGUE  CDRIV2
***PURPOSE  The function of CDRIV2 is to solve N ordinary differential
            equations of the form  $dY(I)/dT = F(Y(I),T)$ , given the
            initial conditions  $Y(I) = YI$ . The program has options to
            allow the solution of both stiff and non-stiff differential
            equations. CDRIV2 allows complex-valued differential
            equations.
***LIBRARY  SLATEC (SDRIVE)
***CATEGORY  I1A2, I1A1B
***TYPE      COMPLEX (SDRIV2-S, DDRIV2-D, CDRIV2-C)
***KEYWORDS  COMPLEX VALUED, GEAR'S METHOD, INITIAL VALUE PROBLEMS,
            ODE, ORDINARY DIFFERENTIAL EQUATIONS, SDRIVE, STIFF
***AUTHOR  Kahaner, D. K., (NIST)
            National Institute of Standards and Technology
            Gaithersburg, MD 20899
            Sutherland, C. D., (LANL)
            Mail Stop D466
            Los Alamos National Laboratory
            Los Alamos, NM 87545
***DESCRIPTION
```

I. PARAMETERS

The user should use parameter names in the call sequence of CDRIV2 for those quantities whose value may be altered by CDRIV2. The parameters in the call sequence are:

N = (Input) The number of differential equations.

T = (Real) The independent variable. On input for the first call, T is the initial point. On output, T is the point at which the solution is given.

Y = (Complex) The vector of dependent variables. Y is used as input on the first call, to set the initial values. On output, Y is the computed solution vector. This array Y is passed in the call sequence of the user-provided routines F and G. Thus parameters required by F and G can be stored in this array in components N+1 and above. (Note: Changes by the user to the first N components of this array will take effect only after a restart, i.e., after setting MSTATE to +1(-1).)

F = A subroutine supplied by the user. The name must be declared EXTERNAL in the user's calling program. This subroutine is of the form:

```
      SUBROUTINE F (N, T, Y, YDOT)
      COMPLEX Y(*), YDOT(*)
```

```
      .
      .
      YDOT(1) = ...
```

```
      .
      .
      YDOT(N) = ...
```

END (Sample)

This computes $YDOT = F(Y,T)$, the right hand side of the differential equations. Here Y is a vector of length at least N . The actual length of Y is determined by the user's declaration in the program which calls CDRIV2. Thus the dimensioning of Y in F , while required by FORTRAN convention, does not actually allocate any storage. When this subroutine is called, the first N components of Y are intermediate approximations to the solution components. The user should not alter these values. Here $YDOT$ is a vector of length N . The user should only compute $YDOT(I)$ for I from 1 to N . Normally a return from F passes control back to CDRIV2. However, if the user would like to abort the calculation, i.e., return control to the program which calls CDRIV2, he should set N to zero. CDRIV2 will signal this by returning a value of $MSTATE$ equal to $+6(-6)$. Altering the value of N in F has no effect on the value of N in the call sequence of CDRIV2.

TOUT = (Input, Real) The point at which the solution is desired.

MSTATE = An integer describing the status of integration. The user must initialize $MSTATE$ to $+1$ or -1 . If $MSTATE$ is positive, the routine will integrate past TOUT and interpolate the solution. This is the most efficient mode. If $MSTATE$ is negative, the routine will adjust its internal step to reach TOUT exactly (useful if a singularity exists beyond TOUT.) The meaning of the magnitude of $MSTATE$:

- 1 (Input) Means the first call to the routine. This value must be set by the user. On all subsequent calls the value of $MSTATE$ should be tested by the user. Unless CDRIV2 is to be reinitialized, only the sign of $MSTATE$ may be changed by the user. (As a convenience to the user who may wish to put out the initial conditions, CDRIV2 can be called with $MSTATE=+1(-1)$, and $TOUT=T$. In this case the program will return with $MSTATE$ unchanged, i.e., $MSTATE=+1(-1)$.)
- 2 (Output) Means a successful integration. If a normal continuation is desired (i.e., a further integration in the same direction), simply advance TOUT and call again. All other parameters are automatically set.
- 3 (Output)(Unsuccessful) Means the integrator has taken 1000 steps without reaching TOUT. The user can continue the integration by simply calling CDRIV2 again. Other than an error in problem setup, the most likely cause for this condition is trying to integrate a stiff set of equations with the non-stiff integrator option. (See description of MINT below.)
- 4 (Output)(Unsuccessful) Means too much accuracy has been requested. EPS has been increased to a value the program estimates is appropriate. The user can continue the integration by simply calling CDRIV2 again.
- 5 (Output) A root was found at a point less than TOUT. The user can continue the integration toward TOUT by simply calling CDRIV2 again.
- 6 (Output)(Unsuccessful) N has been set to zero in SUBROUTINE F .

- 7 (Output)(Unsuccessful) N has been set to zero in FUNCTION G. See description of G below.
- 8 (Output)(Successful) For MSTATE negative, T is beyond TOUT. The solution was obtained by interpolation. The user can continue the integration by simply advancing TOUT and calling CDRIV2 again.
- 9 (Output)(Unsuccessful) The solution could not be obtained. The value of IERFLG (see description below) for a "Recoverable" situation indicates the type of difficulty encountered: either an illegal value for a parameter or an inability to continue the solution. For this condition the user should take corrective action and reset MSTATE to +1(-1) before calling CDRIV2 again. Otherwise the program will terminate the run.

NROOT = (Input) The number of equations whose roots are desired. If NROOT is zero, the root search is not active. This option is useful for obtaining output at points which are not known in advance, but depend upon the solution, e.g., when some solution component takes on a specified value. The root search is carried out using the user-written function G (see description of G below.) CDRIV2 attempts to find the value of T at which one of the equations changes sign. CDRIV2 can find at most one root per equation per internal integration step, and will then return the solution either at TOUT or at a root, whichever occurs first in the direction of integration. The initial point is never reported as a root. The index of the equation whose root is being reported is stored in the sixth element of IWORK.
NOTE: NROOT is never altered by this program.

EPS = (Real) On input, the requested relative accuracy in all solution components. EPS = 0 is allowed. On output, the adjusted relative accuracy if the input value was too small. The value of EPS should be set as large as is reasonable, because the amount of work done by CDRIV2 increases as EPS decreases.

EWT = (Input, Real) Problem zero, i.e., the smallest physically meaningful value for the solution. This is used internally to compute an array YWT(I) = MAX(ABS(Y(I)), EWT). One step error estimates divided by YWT(I) are kept less than EPS. Setting EWT to zero provides pure relative error control. However, setting EWT smaller than necessary can adversely affect the running time.

MINT = (Input) The integration method flag.

- MINT = 1 Means the Adams methods, and is used for non-stiff problems.
- MINT = 2 Means the stiff methods of Gear (i.e., the backward differentiation formulas), and is used for stiff problems.
- MINT = 3 Means the program dynamically selects the Adams methods when the problem is non-stiff and the Gear methods when the problem is stiff.

MINT may not be changed without restarting, i.e., setting the magnitude of MSTATE to 1.

WORK
LENW = (Input)
WORK is an array of LENW complex words used internally for temporary storage. The user must allocate space for this array in the calling program by a statement such as

COMPLEX WORK(...)

The length of WORK should be at least

$16*N + 2*NROOT + 250$ if MINT is 1, or

$N*N + 10*N + 2*NROOT + 250$ if MINT is 2, or

$N*N + 17*N + 2*NROOT + 250$ if MINT is 3,

and LENW should be set to the value used. The contents of WORK should not be disturbed between calls to CDRIV2.

IWORK
LENIW = (Input)
IWORK is an integer array of length LENIW used internally for temporary storage. The user must allocate space for this array in the calling program by a statement such as

INTEGER IWORK(...)

The length of IWORK should be at least

50 if MINT is 1, or

$N+50$ if MINT is 2 or 3,

and LENIW should be set to the value used. The contents of IWORK should not be disturbed between calls to CDRIV2.

G = A real FORTRAN function supplied by the user if NROOT is not 0. In this case, the name must be declared EXTERNAL in the user's calling program. G is repeatedly called with different values of IROOT to obtain the value of each of the NROOT equations for which a root is desired. G is of the form:

REAL FUNCTION G (N, T, Y, IROOT)

COMPLEX Y(*)

GO TO (10, ...), IROOT

10 G = ...

.

.

END (Sample)

Here, Y is a vector of length at least N, whose first N components are the solution components at the point T. The user should not alter these values. The actual length of Y is determined by the user's declaration in the program which calls CDRIV2. Thus the dimensioning of Y in G, while required by FORTRAN convention, does not actually allocate any storage. Normally a return from G passes control back to CDRIV2. However, if the user would like to abort the calculation, i.e., return control to the program which calls CDRIV2, he should set N to zero. CDRIV2 will signal this by returning a value of MSTATE equal to +7(-7). In this case, the index of the equation being evaluated is stored in the sixth element of IWORK. Altering the value of N in G has no effect on the value of N in the call sequence of CDRIV2.

IERFLG = An error flag. The error number associated with a diagnostic message (see Section II-A below) is the same as the corresponding value of IERFLG. The meaning of IERFLG:
0 The routine completed successfully. (No message is

issued.)

3 (Warning) The number of steps required to reach TOUT exceeds MXSTEP.

4 (Warning) The value of EPS is too small.

11 (Warning) For MSTATE negative, T is beyond TOUT. The solution was obtained by interpolation.

15 (Warning) The integration step size is below the roundoff level of T. (The program issues this message as a warning but does not return control to the user.)

22 (Recoverable) N is not positive.

23 (Recoverable) MINT is less than 1 or greater than 3 .

26 (Recoverable) The magnitude of MSTATE is either 0 or greater than 9 .

27 (Recoverable) EPS is less than zero.

32 (Recoverable) Insufficient storage has been allocated for the WORK array.

33 (Recoverable) Insufficient storage has been allocated for the IWORK array.

41 (Recoverable) The integration step size has gone to zero.

42 (Recoverable) The integration step size has been reduced about 50 times without advancing the solution. The problem setup may not be correct.

999 (Fatal) The magnitude of MSTATE is 9 .

II. OTHER COMMUNICATION TO THE USER

- A. The solver communicates to the user through the parameters above. In addition it writes diagnostic messages through the standard error handling program XERMSG. A complete description of XERMSG is given in "Guide to the SLATEC Common Mathematical Library" by Kirby W. Fong et al.. At installations which do not have this error handling package the short but serviceable routine, XERMSG, available with this package, can be used. That program uses the file named OUTPUT to transmit messages.
- B. The first three elements of WORK and the first five elements of IWORK will contain the following statistical data:
- | | |
|--------|---|
| AVGH | The average step size used. |
| HUSED | The step size last used (successfully). |
| AVGORD | The average order used. |
| IMXERR | The index of the element of the solution vector that contributed most to the last error test. |
| NQUSED | The order last used (successfully). |
| NSTEP | The number of steps taken since last initialization. |
| NFE | The number of evaluations of the right hand side. |
| NJE | The number of evaluations of the Jacobian matrix. |

III. REMARKS

- A. On any return from CDRIV2 all information necessary to continue the calculation is contained in the call sequence parameters, including the work arrays. Thus it is possible to suspend one problem, integrate another, and then return to the first.
- B. If this package is to be used in an overlay situation, the user must declare in the primary overlay the variables in the call sequence to CDRIV2.

C. When the routine G is not required, difficulties associated with an unsatisfied external can be avoided by using the name of the routine which calculates the right hand side of the differential equations in place of G in the call sequence of CDRIV2.

IV. USAGE

```

      PROGRAM SAMPLE
      EXTERNAL F
      PARAMETER(MINT = 1, NROOT = 0, N = ...,
8          LENW = 16*N + 2*NROOT + 250, LENIW = 50)
C          N is the number of equations
      COMPLEX WORK(LENW), Y(N)
      REAL EPS, EWT, T, TOUT
      INTEGER IWORK(LENIW)
      OPEN(FILE='TAPE6', UNIT=6, STATUS='NEW')
C
C          Initial point
      T = 0.
C          Set initial conditions
      DO 10 I = 1,N
10      Y(I) = ...
      TOUT = T
      EWT = ...
      MSTATE = 1
      EPS = ...
20      CALL CDRIV2 (N, T, Y, F, TOUT, MSTATE, NROOT, EPS, EWT,
8          MINT, WORK, LENW, IWORK, LENIW, F, IERFLG)
C          Next to last argument is not
C          F if rootfinding is used.
      IF (MSTATE .GT. 2) STOP
      WRITE(6, 100) TOUT, (Y(I), I=1,N)
      TOUT = TOUT + 1.
      IF (TOUT .LE. 10.) GO TO 20
100      FORMAT(...)
      END (Sample)

```

***REFERENCES C. W. Gear, Numerical Initial Value Problems in Ordinary Differential Equations, Prentice-Hall, 1971.

***ROUTINES CALLED CDRIV3, XERMSG

***REVISION HISTORY (YMMDD)

790601 DATE WRITTEN

900329 Initial submission to SLATEC.

END PROLOGUE

CDRIV3

```
SUBROUTINE CDRIV3 (N, T, Y, F, NSTATE, TOUT, NTASK, NROOT, EPS,
8   EWT, IERROR, MINT, MITER, IMPL, ML, MU, MXORD, HMAX, WORK,
8   LENW, IWORK, LENIW, JACOB, FA, NDE, MXSTEP, G, USERS, IERFLG)
***BEGIN PROLOGUE  CDRIV3
***PURPOSE  The function of CDRIV3 is to solve N ordinary differential
            equations of the form  $dY(I)/dT = F(Y(I),T)$ , given the
            initial conditions  $Y(I) = YI$ . The program has options to
            allow the solution of both stiff and non-stiff differential
            equations. Other important options are available. CDRIV3
            allows complex-valued differential equations.
***LIBRARY    SLATEC (SDRIVE)
***CATEGORY   11A2, 11A1B
***TYPE       COMPLEX (SDRIV3-S, DDRIV3-D, CDRIV3-C)
***KEYWORDS   COMPLEX VALUED, GEAR'S METHOD, INITIAL VALUE PROBLEMS,
            ODE, ORDINARY DIFFERENTIAL EQUATIONS, SDRIVE, STIFF
***AUTHOR    Kahaner, D. K., (NIST)
            National Institute of Standards and Technology
            Gaithersburg, MD 20899
            Sutherland, C. D., (LANL)
            Mail Stop D466
            Los Alamos National Laboratory
            Los Alamos, NM 87545
***DESCRIPTION
```

I. ABSTRACT

The primary function of CDRIV3 is to solve N ordinary differential equations of the form $dY(I)/dT = F(Y(I),T)$, given the initial conditions $Y(I) = YI$. The program has options to allow the solution of both stiff and non-stiff differential equations. In addition, CDRIV3 may be used to solve:

1. The initial value problem, $A*dY(I)/dT = F(Y(I),T)$, where A is a non-singular matrix depending on Y and T.
2. The hybrid differential/algebraic initial value problem, $A*dY(I)/dT = F(Y(I),T)$, where A is a vector (whose values may depend upon Y and T) some of whose components will be zero corresponding to those equations which are algebraic rather than differential.

CDRIV3 is to be called once for each output point of T.

II. PARAMETERS

The user should use parameter names in the call sequence of CDRIV3 for those quantities whose value may be altered by CDRIV3. The parameters in the call sequence are:

- N = (Input) The number of dependent functions whose solution is desired. N must not be altered during a problem.
- T = (Real) The independent variable. On input for the first call, T is the initial point. On output, T is the point at which the solution is given.
- Y = (Complex) The vector of dependent variables. Y is used as input on the first call, to set the initial values. On output, Y is the computed solution vector. This array Y

is passed in the call sequence of the user-provided routines F, JACOB, FA, USERS, and G. Thus parameters required by those routines can be stored in this array in components N+1 and above. (Note: Changes by the user to the first N components of this array will take effect only after a restart, i.e., after setting NSTATE to 1.)

F = A subroutine supplied by the user. The name must be declared EXTERNAL in the user's calling program. This subroutine is of the form:

```
SUBROUTINE F (N, T, Y, YDOT)
  COMPLEX Y(*), YDOT(*)
```

```
  .
  .
  YDOT(1) = ...
```

```
  .
  .
  YDOT(N) = ...
END (Sample)
```

This computes $YDOT = F(Y, T)$, the right hand side of the differential equations. Here Y is a vector of length at least N. The actual length of Y is determined by the user's declaration in the program which calls CDRIV3. Thus the dimensioning of Y in F, while required by FORTRAN convention, does not actually allocate any storage. When this subroutine is called, the first N components of Y are intermediate approximations to the solution components. The user should not alter these values. Here YDOT is a vector of length N. The user should only compute YDOT(I) for I from 1 to N. Normally a return from F passes control back to CDRIV3. However, if the user would like to abort the calculation, i.e., return control to the program which calls CDRIV3, he should set N to zero. CDRIV3 will signal this by returning a value of NSTATE equal to 6. Altering the value of N in F has no effect on the value of N in the call sequence of CDRIV3.

NSTATE = An integer describing the status of integration. The meaning of NSTATE is as follows:

- 1 (Input) Means the first call to the routine. This value must be set by the user. On all subsequent calls the value of NSTATE should be tested by the user, but must not be altered. (As a convenience to the user who may wish to put out the initial conditions, CDRIV3 can be called with NSTATE=1, and TOUT=T. In this case the program will return with NSTATE unchanged, i.e., NSTATE=1.)
- 2 (Output) Means a successful integration. If a normal continuation is desired (i.e., a further integration in the same direction), simply advance TOUT and call again. All other parameters are automatically set.
- 3 (Output)(Unsuccessful) Means the integrator has taken MXSTEP steps without reaching TOUT. The user can continue the integration by simply calling CDRIV3 again.
- 4 (Output)(Unsuccessful) Means too much accuracy has been requested. EPS has been increased to a value the program estimates is appropriate. The user can continue the integration by simply calling CDRIV3 again.

- 5 (Output) A root was found at a point less than TOUT. The user can continue the integration toward TOUT by simply calling CDRIV3 again.
- 6 (Output)(Unsuccessful) N has been set to zero in SUBROUTINE F.
- 7 (Output)(Unsuccessful) N has been set to zero in FUNCTION G. See description of G below.
- 8 (Output)(Unsuccessful) N has been set to zero in SUBROUTINE JACOB. See description of JACOB below.
- 9 (Output)(Unsuccessful) N has been set to zero in SUBROUTINE FA. See description of FA below.
- 10 (Output)(Unsuccessful) N has been set to zero in SUBROUTINE USERS. See description of USERS below.
- 11 (Output)(Successful) For NTASK = 2 or 3, T is beyond TOUT. The solution was obtained by interpolation. The user can continue the integration by simply advancing TOUT and calling CDRIV3 again.
- 12 (Output)(Unsuccessful) The solution could not be obtained. The value of IERFLG (see description below) for a "Recoverable" situation indicates the type of difficulty encountered: either an illegal value for a parameter or an inability to continue the solution. For this condition the user should take corrective action and reset NSTATE to 1 before calling CDRIV3 again. Otherwise the program will terminate the run.

TOUT = (Input, Real) The point at which the solution is desired. The position of TOUT relative to T on the first call determines the direction of integration.

NTASK = (Input) An index specifying the manner of returning the solution, according to the following:

- NTASK = 1 Means CDRIV3 will integrate past TOUT and interpolate the solution. This is the most efficient mode.
- NTASK = 2 Means CDRIV3 will return the solution after each internal integration step, or at TOUT, whichever comes first. In the latter case, the program integrates exactly to TOUT.
- NTASK = 3 Means CDRIV3 will adjust its internal step to reach TOUT exactly (useful if a singularity exists beyond TOUT.)

NROOT = (Input) The number of equations whose roots are desired. If NROOT is zero, the root search is not active. This option is useful for obtaining output at points which are not known in advance, but depend upon the solution, e.g., when some solution component takes on a specified value. The root search is carried out using the user-written function G (see description of G below.) CDRIV3 attempts to find the value of T at which one of the equations changes sign. CDRIV3 can find at most one root per equation per internal integration step, and will then return the solution either at TOUT or at a root, whichever occurs first in the direction of integration. The initial point is never reported as a root. The index of the equation whose root is being reported is stored in the sixth element of IWORK.

NOTE: NROOT is never altered by this program.

EPS = (Real) On input, the requested relative accuracy in all solution components. EPS = 0 is allowed. On output, the adjusted relative accuracy if the input value was too small. The value of EPS should be set as large as is reasonable, because the amount of work done by CDRIV3 increases as EPS decreases.

EWT = (Input, Real) Problem zero, i.e., the smallest, nonzero, physically meaningful value for the solution. (Array, possibly of length one. See following description of IERROR.) Setting EWT smaller than necessary can adversely affect the running time.

IERROR = (Input) Error control indicator. A value of 3 is suggested for most problems. Other choices and detailed explanations of EWT and IERROR are given below for those who may need extra flexibility.

These last three input quantities EPS, EWT and IERROR control the accuracy of the computed solution. EWT and IERROR are used internally to compute an array YWT. One step error estimates divided by YWT(I) are kept less than EPS in root mean square norm.

IERROR (Set by the user) =

- 1 Means $YWT(I) = 1$. (Absolute error control)
EWT is ignored.
- 2 Means $YWT(I) = ABS(Y(I))$, (Relative error control)
EWT is ignored.
- 3 Means $YWT(I) = MAX(ABS(Y(I)), EWT(1))$.
- 4 Means $YWT(I) = MAX(ABS(Y(I)), EWT(I))$.
This choice is useful when the solution components have differing scales.
- 5 Means $YWT(I) = EWT(I)$.

If IERROR is 3, EWT need only be dimensioned one.

If IERROR is 4 or 5, the user must dimension EWT at least N, and set its values.

MINT = (Input) The integration method indicator.

- MINT = 1 Means the Adams methods, and is used for non-stiff problems.
- MINT = 2 Means the stiff methods of Gear (i.e., the backward differentiation formulas), and is used for stiff problems.
- MINT = 3 Means the program dynamically selects the Adams methods when the problem is non-stiff and the Gear methods when the problem is stiff. When using the Adams methods, the program uses a value of MITER=0; when using the Gear methods, the program uses the value of MITER provided by the user. Only a value of IMPL = 0 and a value of MITER = 1, 2, 4, or 5 is allowed for this option. The user may not alter the value of MINT or MITER without restarting, i.e., setting NSTATE to 1.

MITER = (Input) The iteration method indicator.

- MITER = 0 Means functional iteration. This value is suggested for non-stiff problems.
- MITER = 1 Means chord method with analytic Jacobian.

In this case, the user supplies subroutine JACOBN (see description below).

MITER = 2 Means chord method with Jacobian calculated internally by finite differences.

MITER = 3 Means chord method with corrections computed by the user-written routine USERS (see description of USERS below.) This option allows all matrix algebra and storage decisions to be made by the user. When using a value of MITER = 3, the subroutine FA is not required, even if IMPL is not 0. For further information on using this option, see Section IV-E below.

MITER = 4 Means the same as MITER = 1 but the A and Jacobian matrices are assumed to be banded.

MITER = 5 Means the same as MITER = 2 but the A and Jacobian matrices are assumed to be banded.

IMPL = (Input) The implicit method indicator.

IMPL = 0 Means solving $dY(I)/dT = F(Y(I),T)$.

IMPL = 1 Means solving $A*dY(I)/dT = F(Y(I),T)$, non-singular A (see description of FA below.) Only MINT = 1 or 2, and MITER = 1, 2, 3, 4, or 5 are allowed for this option.

IMPL = 2,3 Means solving certain systems of hybrid differential/algebraic equations (see description of FA below.) Only MINT = 2 and MITER = 1, 2, 3, 4, or 5, are allowed for this option.

The value of IMPL must not be changed during a problem.

ML = (Input) The lower half-bandwidth in the case of a banded A or Jacobian matrix. (I.e., maximum(R-C) for nonzero A(R,C).)

MU = (Input) The upper half-bandwidth in the case of a banded A or Jacobian matrix. (I.e., maximum(C-R).)

MXORD = (Input) The maximum order desired. This is .LE. 12 for the Adams methods and .LE. 5 for the Gear methods. Normal value is 12 and 5, respectively. If MINT is 3, the maximum order used will be MIN(MXORD, 12) when using the Adams methods, and MIN(MXORD, 5) when using the Gear methods. MXORD must not be altered during a problem.

HMAX = (Input, Real) The maximum magnitude of the step size that will be used for the problem. This is useful for ensuring that important details are not missed. If this is not the case, a large value, such as the interval length, is suggested.

WORK
LENW = (Input)

WORK is an array of LENW complex words used internally for temporary storage. The user must allocate space for this array in the calling program by a statement such as

COMPLEX WORK(...)

The following table gives the required minimum value for the length of WORK, depending on the value of IMPL and

MITER. LENW should be set to the value used. The contents of WORK should not be disturbed between calls to CDRIV3.

IMPL =	0	1	2	3
MITER = 0	(MXORD+4)*N + 2*NROOT + 250	Not allowed	Not allowed	Not allowed
1,2	N*N + (MXORD+5)*N + 2*NROOT + 250	2*N*N + (MXORD+5)*N + 2*NROOT + 250	N*N + (MXORD+6)*N + 2*NROOT + 250	N*(N + NDE) + (MXORD+5)*N + 2*NROOT + 250
3	(MXORD+4)*N + 2*NROOT + 250	(MXORD+4)*N + 2*NROOT + 250	(MXORD+4)*N + 2*NROOT + 250	(MXORD+4)*N + 2*NROOT + 250
4,5	(2*ML+MU+1) *N + (MXORD+5)*N + 2*NROOT + 250	2*(2*ML+MU+1) *N + (MXORD+5)*N + 2*NROOT + 250	(2*ML+MU+1) *N + (MXORD+6)*N + 2*NROOT + 250	(2*ML+MU+1)* (N+NDE) + (MXORD+5)*N + 2*NROOT + 250

IWORK

LENIW = (Input)

IWORK is an integer array of length LENIW used internally for temporary storage. The user must allocate space for this array in the calling program by a statement such as
 INTEGER IWORK(...)

The length of IWORK should be at least

50 if MITER is 0 or 3, or

N+50 if MITER is 1, 2, 4, or 5, or MINT is 3,

and LENIW should be set to the value used. The contents of IWORK should not be disturbed between calls to CDRIV3.

JACOBN = A subroutine supplied by the user, if MITER is 1 or 4.

If this is the case, the name must be declared EXTERNAL in the user's calling program. Given a system of N differential equations, it is meaningful to speak about the partial derivative of the I-th right hand side with respect to the J-th dependent variable. In general there are N*N such quantities. Often however the equations can be ordered so that the I-th differential equation only involves dependent variables with index near I, e.g., I+1, I-2. Such a system is called banded. If, for all I, the I-th equation depends on at most the variables

Y(I-ML), Y(I-ML+1), ..., Y(I), Y(I+1), ..., Y(I+MU)

then we call ML+MU+1 the bandwidth of the system. In a banded system many of the partial derivatives above are automatically zero. For the cases MITER = 1, 2, 4, and 5, some of these partials are needed. For the cases MITER = 2 and 5 the necessary derivatives are approximated numerically by CDRIV3, and we only ask the user to tell CDRIV3 the value of ML and MU if the system is banded. For the cases MITER = 1 and 4 the user must derive these partials algebraically and encode them in subroutine JACOBN. By computing these derivatives the

user can often save 20-30 per cent of the computing time. Usually, however, the accuracy is not much affected and most users will probably forego this option. The optional user-written subroutine JACOBN has the form:

```
SUBROUTINE JACOBN (N, T, Y, DFDY, MATDIM, ML, MU)
  COMPLEX Y(*), DFDY(MATDIM,*)
```

```
  .
  .
  Calculate values of DFDY
```

```
  .
  .
  END (Sample)
```

Here Y is a vector of length at least N. The actual length of Y is determined by the user's declaration in the program which calls CDRIV3. Thus the dimensioning of Y in JACOBN, while required by FORTRAN convention, does not actually allocate any storage. When this subroutine is called, the first N components of Y are intermediate approximations to the solution components. The user should not alter these values. If the system is not banded (MITER=1), the partials of the I-th equation with respect to the J-th dependent function are to be stored in DFDY(I,J). Thus partials of the I-th equation are stored in the I-th row of DFDY. If the system is banded (MITER=4), then the partials of the I-th equation with respect to Y(J) are to be stored in DFDY(K,J), where $K=I-J+MU+1$. Normally a return from JACOBN passes control back to CDRIV3. However, if the user would like to abort the calculation, i.e., return control to the program which calls CDRIV3, he should set N to zero. CDRIV3 will signal this by returning a value of NSTATE equal to +8(-8). Altering the value of N in JACOBN has no effect on the value of N in the call sequence of CDRIV3.

FA = A subroutine supplied by the user if IMPL is not zero, and MITER is not 3. If so, the name must be declared EXTERNAL in the user's calling program. This subroutine computes the array A, where $A \cdot dY(I)/dT = F(Y(I),T)$. There are three cases:

IMPL=1.

Subroutine FA is of the form:

```
SUBROUTINE FA (N, T, Y, A, MATDIM, ML, MU, NDE)
  COMPLEX Y(*), A(MATDIM,*)
```

```
  .
  .
  Calculate ALL values of A
```

```
  .
  .
  END (Sample)
```

In this case A is assumed to be a nonsingular matrix, with the same structure as DFDY (see JACOBN description above). Programming considerations prevent complete generality. If MITER is 1 or 2, A is assumed to be full and the user must compute and store all values of $A(I,J)$, $I,J=1, \dots, N$. If MITER is 4 or 5, A is assumed to be banded with lower and upper half bandwidth ML and MU. The left hand side of the I-th equation is a linear combination of $dY(I-ML)/dT$, $dY(I-ML+1)/dT$, \dots , $dY(I)/dT$, \dots , $dY(I+MU-1)/dT$, $dY(I+MU)/dT$. Thus in the

I-th equation, the coefficient of $dY(J)/dT$ is to be stored in $A(K,J)$, where $K=I-J+MU+1$.
 NOTE: The array A will be altered between calls to FA.

IMPL=2.

Subroutine FA is of the form:

```

      SUBROUTINE FA (N, T, Y, A, MATDIM, ML, MU, NDE)
      COMPLEX Y(*), A(*)

```

```

      .
      .
      Calculate non-zero values of A(1),...,A(NDE)
      .
      .

```

END (Sample)

In this case it is assumed that the system is ordered by the user so that the differential equations appear first, and the algebraic equations appear last. The algebraic equations must be written in the form: $0 = F(Y(I),T)$. When using this option it is up to the user to provide initial values for the $Y(I)$ that satisfy the algebraic equations as well as possible. It is further assumed that A is a vector of length NDE. All of the components of A, which may depend on T, $Y(I)$, etc., must be set by the user to non-zero values.

IMPL=3.

Subroutine FA is of the form:

```

      SUBROUTINE FA (N, T, Y, A, MATDIM, ML, MU, NDE)
      COMPLEX Y(*), A(MATDIM,*)

```

```

      .
      .
      Calculate ALL values of A
      .
      .

```

END (Sample)

In this case A is assumed to be a nonsingular NDE by NDE matrix with the same structure as DFDY (see JACOBN description above). Programming considerations prevent complete generality. If MITER is 1 or 2, A is assumed to be full and the user must compute and store all values of $A(I,J)$, $I,J=1, \dots, NDE$. If MITER is 4 or 5, A is assumed to be banded with lower and upper half bandwidths ML and MU. The left hand side of the I-th equation is a linear combination of $dY(I-ML)/dT$, $dY(I-ML+1)/dT, \dots, dY(I)/dT, \dots, dY(I+MU-1)/dT, dY(I+MU)/dT$. Thus in the I-th equation, the coefficient of $dY(J)/dT$ is to be stored in $A(K,J)$, where $K=I-J+MU+1$. It is assumed that the system is ordered by the user so that the differential equations appear first, and the algebraic equations appear last. The algebraic equations must be written in the form $0 = F(Y(I),T)$. When using this option it is up to the user to provide initial values for the $Y(I)$ that satisfy the algebraic equations as well as possible.

NOTE: For IMPL = 3, the array A will be altered between calls to FA.

Here Y is a vector of length at least N. The actual length of Y is determined by the user's declaration in the program which calls CDRIV3. Thus the dimensioning of Y in FA, while required by FORTRAN convention, does not

actually allocate any storage. When this subroutine is called, the first N components of Y are intermediate approximations to the solution components. The user should not alter these values. FA is always called immediately after calling F, with the same values of T and Y. Normally a return from FA passes control back to CDRIV3. However, if the user would like to abort the calculation, i.e., return control to the program which calls CDRIV3, he should set N to zero. CDRIV3 will signal this by returning a value of NSTATE equal to +9(-9). Altering the value of N in FA has no effect on the value of N in the call sequence of CDRIV3.

NDE = (Input) The number of differential equations. This is required only for IMPL = 2 or 3, with NDE .LT. N.

MXSTEP = (Input) The maximum number of internal steps allowed on one call to CDRIV3.

G = A real FORTRAN function supplied by the user if NROOT is not 0. In this case, the name must be declared EXTERNAL in the user's calling program. G is repeatedly called with different values of IROOT to obtain the value of each of the NROOT equations for which a root is desired. G is of the form:

```

      REAL FUNCTION G (N, T, Y, IROOT)
      COMPLEX Y(*)
      GO TO (10, ...), IROOT
10    G = ...
      .
      .
      END (Sample)

```

Here, Y is a vector of length at least N, whose first N components are the solution components at the point T. The user should not alter these values. The actual length of Y is determined by the user's declaration in the program which calls CDRIV3. Thus the dimensioning of Y in G, while required by FORTRAN convention, does not actually allocate any storage. Normally a return from G passes control back to CDRIV3. However, if the user would like to abort the calculation, i.e., return control to the program which calls CDRIV3, he should set N to zero. CDRIV3 will signal this by returning a value of NSTATE equal to +7(-7). In this case, the index of the equation being evaluated is stored in the sixth element of IWORK. Altering the value of N in G has no effect on the value of N in the call sequence of CDRIV3.

USERS = A subroutine supplied by the user, if MITER is 3. If this is the case, the name must be declared EXTERNAL in the user's calling program. The routine USERS is called by CDRIV3 when certain linear systems must be solved. The user may choose any method to form, store and solve these systems in order to obtain the solution result that is returned to CDRIV3. In particular, this allows sparse matrix methods to be used. The call sequence for this routine is:

```

      SUBROUTINE USERS (Y, YH, YWT, SAVE1, SAVE2, T, H, EL,
8          IMPL, N, NDE, IFLAG)

```

```

COMPLEX Y(*), YH(*), YWT(*), SAVE1(*), SAVE2(*)
REAL T, H, EL

```

The input variable IFLAG indicates what action is to be taken. Subroutine USERS should perform the following operations, depending on the value of IFLAG and IMPL.

```

IFLAG = 0
  IMPL = 0.  USERS is not called.
  IMPL = 1, 2 or 3.  Solve the system  $A \cdot X = \text{SAVE2}$ ,
    returning the result in SAVE2.  The array SAVE1 can
    be used as a work array.  For IMPL = 1, there are N
    components to the system, and for IMPL = 2 or 3,
    there are NDE components to the system.

```

```

IFLAG = 1
  IMPL = 0.  Compute, decompose and store the matrix
     $(I - H \cdot EL \cdot J)$ , where I is the identity matrix and J
    is the Jacobian matrix of the right hand side.  The
    array SAVE1 can be used as a work array.
  IMPL = 1, 2 or 3.  Compute, decompose and store the
    matrix  $(A - H \cdot EL \cdot J)$ .  The array SAVE1 can be used as
    a work array.

```

```

IFLAG = 2
  IMPL = 0.  Solve the system
     $(I - H \cdot EL \cdot J) \cdot X = H \cdot \text{SAVE2} - YH - \text{SAVE1}$ ,
    returning the result in SAVE2.
  IMPL = 1, 2 or 3.  Solve the system
     $(A - H \cdot EL \cdot J) \cdot X = H \cdot \text{SAVE2} - A \cdot (YH + \text{SAVE1})$ 
    returning the result in SAVE2.
  The array SAVE1 should not be altered.
If IFLAG is 0 and IMPL is 1 or 2 and the matrix A is
singular, or if IFLAG is 1 and one of the matrices
 $(I - H \cdot EL \cdot J)$ ,  $(A - H \cdot EL \cdot J)$  is singular, the INTEGER
variable IFLAG is to be set to -1 before RETURNing.
Normally a return from USERS passes control back to
CDRIV3.  However, if the user would like to abort the
calculation, i.e., return control to the program which
calls CDRIV3, he should set N to zero.  CDRIV3 will signal
this by returning a value of NSTATE equal to +10(-10).
Altering the value of N in USERS has no effect on the
value of N in the call sequence of CDRIV3.

```

IERFLG = An error flag. The error number associated with a diagnostic message (see Section III-A below) is the same as the corresponding value of IERFLG. The meaning of IERFLG:

- 0 The routine completed successfully. (No message is issued.)
- 3 (Warning) The number of steps required to reach TOUT exceeds MXSTEP.
- 4 (Warning) The value of EPS is too small.
- 11 (Warning) For NTASK = 2 or 3, T is beyond TOUT. The solution was obtained by interpolation.
- 15 (Warning) The integration step size is below the roundoff level of T. (The program issues this message as a warning but does not return control to the user.)
- 22 (Recoverable) N is not positive.

23 (Recoverable) MINT is less than 1 or greater than 3 .
 24 (Recoverable) MITER is less than 0 or greater than
 5 .
 25 (Recoverable) IMPL is less than 0 or greater than 3 .
 26 (Recoverable) The value of NSTATE is less than 1 or
 greater than 12 .
 27 (Recoverable) EPS is less than zero.
 28 (Recoverable) MXORD is not positive.
 29 (Recoverable) For MINT = 3, either MITER = 0 or 3, or
 IMPL = 0 .
 30 (Recoverable) For MITER = 0, IMPL is not 0 .
 31 (Recoverable) For MINT = 1, IMPL is 2 or 3 .
 32 (Recoverable) Insufficient storage has been allocated
 for the WORK array.
 33 (Recoverable) Insufficient storage has been allocated
 for the IWORK array.
 41 (Recoverable) The integration step size has gone
 to zero.
 42 (Recoverable) The integration step size has been
 reduced about 50 times without advancing the
 solution. The problem setup may not be correct.
 43 (Recoverable) For IMPL greater than 0, the matrix A
 is singular.
 999 (Fatal) The value of NSTATE is 12 .

III. OTHER COMMUNICATION TO THE USER

- A. The solver communicates to the user through the parameters above. In addition it writes diagnostic messages through the standard error handling program XERMSG. A complete description of XERMSG is given in "Guide to the SLATEC Common Mathematical Library" by Kirby W. Fong et al.. At installations which do not have this error handling package the short but serviceable routine, XERMSG, available with this package, can be used. That program uses the file named OUTPUT to transmit messages.
- B. The first three elements of WORK and the first five elements of IWORK will contain the following statistical data:
- | | |
|--------|---|
| AVGH | The average step size used. |
| HUSED | The step size last used (successfully). |
| AVGORD | The average order used. |
| IMXERR | The index of the element of the solution vector that contributed most to the last error test. |
| NQUSED | The order last used (successfully). |
| NSTEP | The number of steps taken since last initialization. |
| NFE | The number of evaluations of the right hand side. |
| NJE | The number of evaluations of the Jacobian matrix. |

IV. REMARKS

- A. Other routines used:
 CDNTP, CDZRO, CDSTP, CDNTL, CDPST, CDCOR, CDCST,
 CDPSC, and CDSCL;
 CGEFA, CGESL, CGBFA, CGBSL, and SCNRM2 (from LINPACK)
 R1MACH (from the Bell Laboratories Machine Constants Package)
 XERMSG (from the SLATEC Common Math Library)
 The last seven routines above, not having been written by the present authors, are not explicitly part of this package.

- B. On any return from CDRIV3 all information necessary to continue
SLATEC2 (AAAAAA through D9UPAK) - 195

the calculation is contained in the call sequence parameters, including the work arrays. Thus it is possible to suspend one problem, integrate another, and then return to the first.

- C. If this package is to be used in an overlay situation, the user must declare in the primary overlay the variables in the call sequence to CDRIV3.
- D. Changing parameters during an integration.
The value of NROOT, EPS, EWT, IERROR, MINT, MITER, or HMAX may be altered by the user between calls to CDRIV3. For example, if too much accuracy has been requested (the program returns with NSTATE = 4 and an increased value of EPS) the user may wish to increase EPS further. In general, prudence is necessary when making changes in parameters since such changes are not implemented until the next integration step, which is not necessarily the next call to CDRIV3. This can happen if the program has already integrated to a point which is beyond the new point TOUT.
- E. As the price for complete control of matrix algebra, the CDRIV3 USERS option puts all responsibility for Jacobian matrix evaluation on the user. It is often useful to approximate numerically all or part of the Jacobian matrix. However this must be done carefully. The FORTRAN sequence below illustrates the method we recommend. It can be inserted directly into subroutine USERS to approximate Jacobian elements in rows I1 to I2 and columns J1 to J2.

```

COMPLEX DFDY(N,N), R, SAVE1(N), SAVE2(N), Y(N), YJ, YWT(N)
REAL EPSJ, H, R1MACH, T, UROUND
UROUND = R1MACH(4)
EPSJ = SQRT(UROUND)
DO 30 J = J1,J2
  IF (ABS(Y(J)) .GT. ABS(YWT(J))) THEN
    R = EPSJ*Y(J)
  ELSE
    R = EPSJ*YWT(J)
  END IF
  IF (R .EQ. 0.E0) R = YWT(J)
  YJ = Y(J)
  Y(J) = Y(J) + R
  CALL F (N, T, Y, SAVE1)
  IF (N .EQ. 0) RETURN
  Y(J) = YJ
DO 20 I = I1,I2
  20   DFDY(I,J) = (SAVE1(I) - SAVE2(I))/R
  30   CONTINUE

```

Many problems give rise to structured sparse Jacobians, e.g., block banded. It is possible to approximate them with fewer function evaluations than the above procedure uses; see Curtis, Powell and Reid, J. Inst. Maths Applics, (1974), Vol. 13, pp. 117-119.

- F. When any of the routines JACOB, FA, G, or USERS, is not required, difficulties associated with unsatisfied externals can be avoided by using the name of the routine which calculates the right hand side of the differential equations in place of the corresponding name in the call sequence of CDRIV3.

Ordinary Differential Equations, Prentice-Hall, 1971.
***ROUTINES CALLED CDNTP, CDSTP, CDZRO, CGBFA, CGBSL, CGEFA, CGESL,
R1MACH, SCNRM2, XERMSG
***REVISION HISTORY (YYMMDD)
790601 DATE WRITTEN
900329 Initial submission to SLATEC.
END PROLOGUE

CEXPRL

```
      COMPLEX FUNCTION CEXPRL (Z)
***BEGIN PROLOGUE  CEXPRL
***PURPOSE  Calculate the relative error exponential (EXP(X)-1)/X.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY   C4B
***TYPE       COMPLEX (EXPREL-S, DEXPRL-D, CEXPRL-C)
***KEYWORDS  ELEMENTARY FUNCTIONS, EXPONENTIAL, FIRST ORDER, FNLIB
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION

      Evaluate (EXP(Z)-1)/Z .  For small ABS(Z), we use the Taylor
      series.  We could instead use the expression
          CEXPRL(Z) = (EXP(X)*EXP(I*Y)-1)/Z
                    = (X*EXPREL(X) * (1 - 2*SIN(Y/2)**2) - 2*SIN(Y/2)**2
                      + I*SIN(Y)*(1+X*EXPREL(X))) / Z

***REFERENCES  (NONE)
***ROUTINES CALLED  R1MACH
***REVISION HISTORY  (YYMMDD)
      770801  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890531  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      END PROLOGUE
```

CFFTB1

```
      SUBROUTINE CFFTB1 (N, C, CH, WA, IFAC)
***BEGIN PROLOGUE  CFFTB1
***PURPOSE  Compute the unnormalized inverse of CFFTF1.
***LIBRARY   SLATEC (FFTPACK)
***CATEGORY  J1A2
***TYPE      COMPLEX (RFFTB1-S, CFFTB1-C)
***KEYWORDS  FFTPACK, FOURIER TRANSFORM
***AUTHOR   Swarztrauber, P. N., (NCAR)
***DESCRIPTION
```

Subroutine CFFTB1 computes the backward complex discrete Fourier transform (the Fourier synthesis). Equivalently, CFFTB1 computes a complex periodic sequence from its Fourier coefficients. The transform is defined below at output parameter C.

A call of CFFTF1 followed by a call of CFFTB1 will multiply the sequence by N.

The arrays WA and IFAC which are used by subroutine CFFTB1 must be initialized by calling subroutine CFFTI1 (N, WA, IFAC).

Input Parameters

N the length of the complex sequence C. The method is more efficient when N is the product of small primes.

C a complex array of length N which contains the sequence

CH a real work array of length at least 2*N

WA a real work array which must be dimensioned at least 2*N.

IFAC an integer work array which must be dimensioned at least 15.

The WA and IFAC arrays must be initialized by calling subroutine CFFTI1 (N, WA, IFAC), and different WA and IFAC arrays must be used for each different value of N. This initialization does not have to be repeated so long as N remains unchanged. Thus subsequent transforms can be obtained faster than the first. The same WA and IFAC arrays can be used by CFFTF1 and CFFTB1.

Output Parameters

C For $J=1, \dots, N$

$C(J) = \text{the sum from } K=1, \dots, N \text{ of}$

$C(K) * \text{EXP}(I * (J-1) * (K-1) * 2 * \text{PI} / N)$

where $I = \text{SQRT}(-1)$

NOTE: WA and IFAC contain initialization calculations which must not be destroyed between calls of subroutine CFFTF1 or CFFTB1

```
***REFERENCES  P. N. Swarztrauber, Vectorizing the FFTs, in Parallel
```

Computations (G. Rodrigue, ed.), Academic Press,
 1982, pp. 51-83.
 ***ROUTINES CALLED PASSB, PASSB2, PASSB3, PASSB4, PASSB5
 ***REVISION HISTORY (YYMMDD)
 790601 DATE WRITTEN
 830401 Modified to use SLATEC library source file format.
 860115 Modified by Ron Boisvert to adhere to Fortran 77 by
 changing dummy array size declarations (1) to (*).
 881128 Modified by Dick Valent to meet prologue standards.
 891214 Prologue converted to Version 4.0 format. (BAB)
 900131 Routine changed from subsidiary to user-callable. (WRB)
 920501 Reformatted the REFERENCES section. (WRB)
 END PROLOGUE

CFFTF1

```
SUBROUTINE CFFTF1 (N, C, CH, WA, IFAC)
***BEGIN PROLOGUE  CFFTF1
***PURPOSE  Compute the forward transform of a complex, periodic
            sequence.
***LIBRARY    SLATEC (FFTPACK)
***CATEGORY   J1A2
***TYPE       COMPLEX (RFFTF1-S, CFFTF1-C)
***KEYWORDS   FFTPAC, FOURIER TRANSFORM
***AUTHOR    Swarztrauber, P. N., (NCAR)
***DESCRIPTION
```

Subroutine CFFTF1 computes the forward complex discrete Fourier transform (the Fourier analysis). Equivalently, CFFTF1 computes the Fourier coefficients of a complex periodic sequence. The transform is defined below at output parameter C.

The transform is not normalized. To obtain a normalized transform the output must be divided by N. Otherwise a call of CFFTF1 followed by a call of CFFTB1 will multiply the sequence by N.

The arrays WA and IFAC which are used by subroutine CFFTB1 must be initialized by calling subroutine CFFTI1 (N, WA, IFAC).

Input Parameters

N the length of the complex sequence C. The method is more efficient when N is the product of small primes.

C a complex array of length N which contains the sequence

CH a real work array of length at least 2*N

WA a real work array which must be dimensioned at least 2*N.

IFAC an integer work array which must be dimensioned at least 15.

The WA and IFAC arrays must be initialized by calling subroutine CFFTI1 (N, WA, IFAC), and different WA and IFAC arrays must be used for each different value of N. This initialization does not have to be repeated so long as N remains unchanged. Thus subsequent transforms can be obtained faster than the first. The same WA and IFAC arrays can be used by CFFTF1 and CFFTB1.

Output Parameters

C For $J=1, \dots, N$

$$C(J) = \text{the sum from } K=1, \dots, N \text{ of}$$
$$C(K) * \exp(-I * (J-1) * (K-1) * 2 * \pi / N)$$

where $I = \text{SQRT}(-1)$

NOTE: WA and IFAC contain initialization calculations which must not be destroyed between calls of subroutine CFFTF1 or CFFTB1

***REFERENCES P. N. Swarztrauber, Vectorizing the FFTs, in Parallel
Computations (G. Rodrigue, ed.), Academic Press,
1982, pp. 51-83.

***ROUTINES CALLED PASSF, PASSF2, PASSF3, PASSF4, PASSF5

***REVISION HISTORY (YYMMDD)

790601 DATE WRITTEN

830401 Modified to use SLATEC library source file format.

860115 Modified by Ron Boisvert to adhere to Fortran 77 by
changing dummy array size declarations (1) to (*).

881128 Modified by Dick Valent to meet prologue standards.

891214 Prologue converted to Version 4.0 format. (BAB)

900131 Routine changed from subsidiary to user-callable. (WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CFFTI

```

      SUBROUTINE CFFTI (N, WSAVE)
***BEGIN PROLOGUE  CFFTI
***SUBSIDIARY
***PURPOSE  Initialize a work array for CFFTF and CFFTB.
***LIBRARY   SLATEC (FFTPACK)
***CATEGORY  J1A2
***TYPE      COMPLEX (RFFTI-S, CFFTI-C)
***KEYWORDS  FFTPACK, FOURIER TRANSFORM
***AUTHOR   Swarztrauber, P. N., (NCAR)
***DESCRIPTION

*****
*   NOTICE   NOTICE   NOTICE   NOTICE   NOTICE   NOTICE   NOTICE   *
*****
*
*   This routine uses non-standard Fortran 77 constructs and will   *
*   be removed from the library at a future date.  You are         *
*   requested to use CFFTI1.                                         *
*
*****

Subroutine CFFTI initializes the array WSAVE which is used in
both CFFTF and CFFTB.  The prime factorization of N together with
a tabulation of the trigonometric functions are computed and
stored in WSAVE.

Input Parameter

N          the length of the sequence to be transformed

Output Parameter

WSAVE      a work array which must be dimensioned at least 4*N+15.
           The same work array can be used for both CFFTF and CFFTB
           as long as N remains unchanged.  Different WSAVE arrays
           are required for different values of N.  The contents of
           WSAVE must not be changed between calls of CFFTF or CFFTB.

***REFERENCES  P. N. Swarztrauber, Vectorizing the FFTs, in Parallel
               Computations (G. Rodrigue, ed.), Academic Press,
               1982, pp. 51-83.
***ROUTINES CALLED  CFFTI1
***REVISION HISTORY  (YYMMDD)
   790601  DATE WRITTEN
   830401  Modified to use SLATEC library source file format.
   860115  Modified by Ron Boisvert to adhere to Fortran 77 by
           changing dummy array size declarations (1) to (*).
   861211  REVISION DATE from Version 3.2
   881128  Modified by Dick Valent to meet prologue standards.
   891214  Prologue converted to Version 4.0 format.  (BAB)
   900131  Routine changed from user-callable to subsidiary
           because of non-standard Fortran 77 arguments in the
           call to CFFTB1.  (WRB)
   920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE

```

CFFT11

```
SUBROUTINE CFFT11 (N, WA, IFAC)
***BEGIN PROLOGUE  CFFT11
***PURPOSE  Initialize a real and an integer work array for CFFTf1 and
             CFFTb1.
***LIBRARY   SLATEC (FFTPACK)
***CATEGORY  J1A2
***TYPE      COMPLEX (RFFT11-S, CFFT11-C)
***KEYWORDS  FFFPACK, FOURIER TRANSFORM
***AUTHOR   Swarztrauber, P. N., (NCAR)
***DESCRIPTION
```

Subroutine CFFT11 initializes the work arrays WA and IFAC which are used in both CFFTf1 and CFFTb1. The prime factorization of N and a tabulation of the trigonometric functions are computed and stored in IFAC and WA, respectively.

Input Parameter

N the length of the sequence to be transformed

Output Parameters

WA a real work array which must be dimensioned at least 2*N.

IFAC an integer work array which must be dimensioned at least 15.

The same work arrays can be used for both CFFTf1 and CFFTb1 as long as N remains unchanged. Different WA and IFAC arrays are required for different values of N. The contents of WA and IFAC must not be changed between calls of CFFTf1 or CFFTb1.

```
***REFERENCES  P. N. Swarztrauber, Vectorizing the FFTs, in Parallel
                Computations (G. Rodrigue, ed.), Academic Press,
                1982, pp. 51-83.
```

```
***ROUTINES CALLED  (NONE)
```

```
***REVISION HISTORY  (YYMMDD)
```

```
790601  DATE WRITTEN
```

```
830401  Modified to use SLATEC library source file format.
```

```
860115  Modified by Ron Boisvert to adhere to Fortran 77 by
        (a) changing dummy array size declarations (1) to (*),
        (b) changing references to intrinsic function FLOAT
            to REAL, and
        (c) changing definition of variable TPI by using
            FORTRAN intrinsic function ATAN instead of a DATA
            statement.
```

```
881128  Modified by Dick Valent to meet prologue standards.
```

```
890531  Changed all specific intrinsics to generic.  (WRB)
```

```
891214  Prologue converted to Version 4.0 format.  (BAB)
```

```
900131  Routine changed from subsidiary to user-callable.  (WRB)
```

```
920501  Reformatted the REFERENCES section.  (WRB)
```

```
END PROLOGUE
```

CG

```
      SUBROUTINE CG (NM, N, AR, AI, WR, WI, MATZ, ZR, ZI, FV1, FV2, FV3,  
+      IERR)  
***BEGIN PROLOGUE  CG  
***PURPOSE  Compute the eigenvalues and, optionally, the eigenvectors  
            of a complex general matrix.  
***LIBRARY   SLATEC (EISPACK)  
***CATEGORY  D4A4  
***TYPE      COMPLEX (RG-S, CG-C)  
***KEYWORDS  EIGENVALUES, EIGENVECTORS, EISPACK  
***AUTHOR   Smith, B. T., et al.  
***DESCRIPTION
```

This subroutine calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPACK) to find the eigenvalues and eigenvectors (if desired) of a COMPLEX GENERAL matrix.

On INPUT

NM must be set to the row dimension of the two-dimensional array parameters, AR, AI, ZR and ZI, as declared in the calling program dimension statement. NM is an INTEGER variable.

N is the order of the matrix A=(AR,AI). N is an INTEGER variable. N must be less than or equal to NM.

AR and AI contain the real and imaginary parts, respectively, of the complex general matrix. AR and AI are two-dimensional REAL arrays, dimensioned AR(NM,N) and AI(NM,N).

MATZ is an INTEGER variable set equal to zero if only eigenvalues are desired. Otherwise, it is set to any non-zero integer for both eigenvalues and eigenvectors.

On OUTPUT

WR and WI contain the real and imaginary parts, respectively, of the eigenvalues. WR and WI are one-dimensional REAL arrays, dimensioned WR(N) and WI(N).

ZR and ZI contain the real and imaginary parts, respectively, of the eigenvectors if MATZ is not zero. ZR and ZI are two-dimensional REAL arrays, dimensioned ZR(NM,N) and ZI(NM,N).

IERR is an INTEGER flag set to
Zero for normal return,
10*N if N is greater than NM,
J if the J-th eigenvalue has not been
 determined after a total of 30 iterations.
The eigenvalues should be correct for indices
IERR+1, IERR+2, ..., N, but no eigenvectors are
computed.

FV1, FV2, and FV3 are one-dimensional REAL arrays used for

temporary storage, dimensioned FV1(N), FV2(N), and FV3(N).

Questions and comments should be directed to B. S. Garbow,
APPLIED MATHEMATICS DIVISION, ARGONNE NATIONAL LABORATORY

***REFERENCES B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow,
Y. Ikebe, V. C. Klema and C. B. Moler, Matrix Eigen-
system Routines - EISPACK Guide, Springer-Verlag,
1976.

***ROUTINES CALLED CBABK2, CBAL, COMQR, COMQR2, CORTH

***REVISION HISTORY (YYMMDD)

760101 DATE WRITTEN

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CGAMMA

```
      COMPLEX FUNCTION CGAMMA (Z)
***BEGIN PROLOGUE  CGAMMA
***PURPOSE  Compute the complete Gamma function.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C7A
***TYPE      COMPLEX (GAMMA-S, DGAMMA-D, CGAMMA-C)
***KEYWORDS  COMPLETE GAMMA FUNCTION, FNLIB, SPECIAL FUNCTIONS
***AUTHOR   Fullerton, W., (LANL)
***DESCRIPTION
```

CGAMMA(Z) calculates the complete gamma function for COMPLEX argument Z. This is a preliminary version that is portable but not accurate.

```
***REFERENCES  (NONE)
***ROUTINES CALLED  CLNGAM
***REVISION HISTORY  (YYMMDD)
    770701  DATE WRITTEN
    861211  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    END PROLOGUE
```

CGAMR

```
      COMPLEX FUNCTION CGAMR (Z)
***BEGIN PROLOGUE  CGAMR
***PURPOSE  Compute the reciprocal of the Gamma function.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C7A
***TYPE      COMPLEX (GAMR-S, DGAMR-D, CGAMR-C)
***KEYWORDS  FNLIB, RECIPROCAL GAMMA FUNCTION, SPECIAL FUNCTIONS
***AUTHOR   Fullerton, W., (LANL)
***DESCRIPTION
```

CGAMR(Z) calculates the reciprocal gamma function for COMPLEX argument Z. This is a preliminary version that is not accurate.

```
***REFERENCES  (NONE)
***ROUTINES CALLED  CLNGAM, XERCLR, XGETF, XSETF
***REVISION HISTORY  (YYMMDD)
      770701  DATE WRITTEN
      861211  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      END PROLOGUE
```

CGBCO

```
      SUBROUTINE CGBCO (ABD, LDA, N, ML, MU, IPVT, RCOND, Z)
***BEGIN PROLOGUE  CGBCO
***PURPOSE  Factor a band matrix by Gaussian elimination and
            estimate the condition number of the matrix.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2C2
***TYPE      COMPLEX (SGBCO-S, DGBCO-D, CGBCO-C)
***KEYWORDS  BANDED, CONDITION NUMBER, LINEAR ALGEBRA, LINPACK,
            MATRIX FACTORIZATION
***AUTHOR  Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CGBCO factors a complex band matrix by Gaussian elimination and estimates the condition of the matrix.

If RCOND is not needed, CGBFA is slightly faster.
To solve $A \cdot X = B$, follow CGBCO by CGBSL.
To compute $\text{INVERSE}(A) \cdot C$, follow CGBCO by CGBSL.
To compute $\text{DETERMINANT}(A)$, follow CGBCO by CGBDI.

On Entry

ABD COMPLEX(LDA, N)
 contains the matrix in band storage. The columns
 of the matrix are stored in the columns of ABD and
 the diagonals of the matrix are stored in rows
 ML+1 through 2*ML+MU+1 of ABD.
 See the comments below for details.

LDA INTEGER
 the leading dimension of the array ABD.
 LDA must be .GE. 2*ML + MU + 1.

N INTEGER
 the order of the original matrix.

ML INTEGER
 number of diagonals below the main diagonal.
 0 .LE. ML .LT. N.

MU INTEGER
 number of diagonals above the main diagonal.
 0 .LE. MU .LT. N.
 More efficient if ML .LE. MU.

On Return

ABD an upper triangular matrix in band storage and
 the multipliers which were used to obtain it.
 The factorization can be written $A = L \cdot U$ where
 L is a product of permutation and unit lower
 triangular matrices and U is upper triangular.

IPVT INTEGER(N)
 an integer vector of pivot indices.

RCOND REAL
 an estimate of the reciprocal condition of A .
 For the system $A \cdot X = B$, relative perturbations
 in A And B of size EPSILON may cause
 relative perturbations in X of size EPSILON/RCOND .
 If RCOND is so small that the logical expression
 $1.0 + RCOND .EQ. 1.0$
 is true, then A may be singular to working
 precision. In particular, RCOND is zero if
 exact singularity is detected or the estimate
 underflows.

Z COMPLEX(N)
 a work vector whose contents are usually unimportant.
 If A is close to a singular matrix, then Z is
 an approximate null vector in the sense that
 $NORM(A \cdot Z) = RCOND \cdot NORM(A) \cdot NORM(Z)$.

Band Storage

if A is a band matrix, the following program segment
 will set up the input.

```

      ML = (band width below the diagonal)
      MU = (band width above the diagonal)
      M = ML + MU + 1
      DO 20 J = 1, N
        I1 = MAX(1, J-MU)
        I2 = MIN(N, J+ML)
        DO 10 I = I1, I2
          K = I - J + M
          ABD(K,J) = A(I,J)
        10 CONTINUE
      20 CONTINUE

```

This uses rows ML+1 through 2*ML+MU+1 of ABD .
 In addition, the first ML rows in ABD are used for
 elements generated during the triangularization.
 The total number of rows needed in ABD is 2*ML+MU+1 .
 The ML+MU by ML+MU upper left triangle and the
 ML by ML lower right triangle are not referenced.

Example: If the original matrix is

```

11 12 13  0  0  0
21 22 23 24  0  0
  0 32 33 34 35  0
  0  0 43 44 45 46
  0  0  0 54 55 56
  0  0  0  0 65 66

```

then N = 6, ML = 1, MU = 2, LDA .GE. 5 and ABD should contain

```

*  *  *  +  +  +  , * = not used
*  * 13 24 35 46  , + = used for pivoting
* 12 23 34 45 56
11 22 33 44 55 66
21 32 43 54 65  *

```

```

                Stewart, LINPACK Users' Guide, SIAM, 1979.
***ROUTINES CALLED  CAXPY, CDOTC, CGBFA, CSSCAL, SCASUM
***REVISION HISTORY  (YYMMDD)
  780814  DATE WRITTEN
  890531  Changed all specific intrinsics to generic.  (WRB)
  890831  Modified array declarations.  (WRB)
  890831  REVISION DATE from Version 3.2
  891214  Prologue converted to Version 4.0 format.  (BAB)
  900326  Removed duplicate information from DESCRIPTION section.
          (WRB)
  920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE

```

CGBDI

```
      SUBROUTINE CGBDI (ABD, LDA, N, ML, MU, IPVT, DET)
***BEGIN PROLOGUE  CGBDI
***PURPOSE  Compute the determinant of a complex band matrix using the
             factors from CGBCO or CGBFA.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D3C2
***TYPE      COMPLEX (SGBDI-S, DGBDI-D, CGBDI-C)
***KEYWORDS  BANDED, DETERMINANT, INVERSE, LINEAR ALGEBRA, LINPACK,
             MATRIX
***AUTHOR    Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CGBDI computes the determinant of a band matrix using the factors computed by CGBCO or CGBFA. If the inverse is needed, use CGBSL N times.

On Entry

ABD COMPLEX(LDA, N)
 the output from CGBCO or CGBFA.

LDA INTEGER
 the leading dimension of the array ABD .

N INTEGER
 the order of the original matrix.

ML INTEGER
 number of diagonals below the main diagonal.

MU INTEGER
 number of diagonals above the main diagonal.

IPVT INTEGER(N)
 the pivot vector from CGBCO or CGBFA.

On Return

DET COMPLEX(2)
 determinant of original matrix.
 Determinant = DET(1) * 10.0**DET(2)
 with 1.0 .LE. CABS1(DET(1)) .LT. 10.0
 or DET(1) = 0.0 .

```
***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
               Stewart, LINPACK Users' Guide, SIAM, 1979.
```

```
***ROUTINES CALLED  (NONE)
```

```
***REVISION HISTORY  (YMMDD)
```

```
780814  DATE WRITTEN
```

```
890831  Modified array declarations.  (WRB)
```

```
890831  REVISION DATE from Version 3.2
```

```
891214  Prologue converted to Version 4.0 format.  (BAB)
```

```
900326  Removed duplicate information from DESCRIPTION section.
       (WRB)
```

```
920501  Reformatted the REFERENCES section.  (WRB)
```

```
END PROLOGUE
```

CGBFA

```
SUBROUTINE CGBFA (ABD, LDA, N, ML, MU, IPVT, INFO)
***BEGIN PROLOGUE  CGBFA
***PURPOSE  Factor a band matrix using Gaussian elimination.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2C2
***TYPE      COMPLEX (SGBFA-S, DGBFA-D, CGBFA-C)
***KEYWORDS  BANDED, LINEAR ALGEBRA, LINPACK, MATRIX FACTORIZATION
***AUTHOR  Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CGBFA factors a complex band matrix by elimination.

CGBFA is usually called by CGBCO, but it can be called directly with a saving in time if RCOND is not needed.

On Entry

ABD COMPLEX(LDA, N)
 contains the matrix in band storage. The columns
 of the matrix are stored in the columns of ABD and
 the diagonals of the matrix are stored in rows
 ML+1 through 2*ML+MU+1 of ABD .
 See the comments below for details.

LDA INTEGER
 the leading dimension of the array ABD .
 LDA must be .GE. 2*ML + MU + 1 .

N INTEGER
 the order of the original matrix.

ML INTEGER
 number of diagonals below the main diagonal.
 0 .LE. ML .LT. N .

MU INTEGER
 number of diagonals above the main diagonal.
 0 .LE. MU .LT. N .
 More efficient if ML .LE. MU .

On Return

ABD an upper triangular matrix in band storage and
 the multipliers which were used to obtain it.
 The factorization can be written $A = L*U$ where
 L is a product of permutation and unit lower
 triangular matrices and U is upper triangular.

IPVT INTEGER(N)
 an integer vector of pivot indices.

INFO INTEGER
 = 0 normal value.
 = K if U(K,K) .EQ. 0.0 . This is not an error
 condition for this subroutine, but it does
 indicate that CGBSL will divide by zero if
 called. Use RCOND in CGBCO for a reliable

indication of singularity.

Band Storage

If A is a band matrix, the following program segment will set up the input.

```
      ML = (band width below the diagonal)
      MU = (band width above the diagonal)
      M = ML + MU + 1
      DO 20 J = 1, N
        I1 = MAX(1, J-MU)
        I2 = MIN(N, J+ML)
        DO 10 I = I1, I2
          K = I - J + M
          ABD(K,J) = A(I,J)
        10 CONTINUE
      20 CONTINUE
```

This uses rows ML+1 through 2*ML+MU+1 of ABD .
In addition, the first ML rows in ABD are used for
elements generated during the triangularization.
The total number of rows needed in ABD is 2*ML+MU+1 .
The ML+MU by ML+MU upper left triangle and the
ML by ML lower right triangle are not referenced.

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CAXPY, CSCAL, ICAMAX

***REVISION HISTORY (YYMMDD)

780814 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900326 Removed duplicate information from DESCRIPTION section.
(WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CGBMV

```
      SUBROUTINE CGBMV (TRANS, M, N, KL, KU, ALPHA, A, LDA, X, INCX,  
$      BETA, Y, INCY)  
***BEGIN PROLOGUE  CGBMV  
***PURPOSE  Multiply a complex vector by a complex general band matrix.  
***LIBRARY   SLATEC (BLAS)  
***CATEGORY  D1B4  
***TYPE      COMPLEX (SGBMV-S, DGBMV-D, CGBMV-C)  
***KEYWORDS  LEVEL 2 BLAS, LINEAR ALGEBRA  
***AUTHOR   Dongarra, J. J., (ANL)  
            Du Croz, J., (NAG)  
            Hammarling, S., (NAG)  
            Hanson, R. J., (SNLA)  
***DESCRIPTION
```

CGBMV performs one of the matrix-vector operations

$$y := \alpha A x + \beta y, \quad \text{or} \quad y := \alpha A' x + \beta y, \quad \text{or}$$
$$y := \alpha \text{conjg}(A') x + \beta y,$$

where α and β are scalars, x and y are vectors and A is an m by n band matrix, with kl sub-diagonals and ku super-diagonals.

Parameters
=====

TRANS - CHARACTER*1.
On entry, TRANS specifies the operation to be performed as follows:

TRANS = 'N' or 'n' $y := \alpha A x + \beta y$.

TRANS = 'T' or 't' $y := \alpha A' x + \beta y$.

TRANS = 'C' or 'c' $y := \alpha \text{conjg}(A') x + \beta y$.

Unchanged on exit.

M - INTEGER.
On entry, M specifies the number of rows of the matrix A.
M must be at least zero.
Unchanged on exit.

N - INTEGER.
On entry, N specifies the number of columns of the matrix A.
N must be at least zero.
Unchanged on exit.

KL - INTEGER.
On entry, KL specifies the number of sub-diagonals of the matrix A. KL must satisfy $0 \leq KL$.
Unchanged on exit.

KU - INTEGER.
On entry, KU specifies the number of super-diagonals of the matrix A. KU must satisfy $0 \leq KU$.

Unchanged on exit.

ALPHA - COMPLEX .
On entry, ALPHA specifies the scalar alpha.
Unchanged on exit.

A - COMPLEX array of DIMENSION (LDA, n).
Before entry, the leading (kl + ku + 1) by n part of the array A must contain the matrix of coefficients, supplied column by column, with the leading diagonal of the matrix in row (ku + 1) of the array, the first super-diagonal starting at position 2 in row ku, the first sub-diagonal starting at position 1 in row (ku + 2), and so on. Elements in the array A that do not correspond to elements in the band matrix (such as the top left ku by ku triangle) are not referenced.
The following program segment will transfer a band matrix from conventional full matrix storage to band storage:

```
      DO 20, J = 1, N
        K = KU + 1 - J
        DO 10, I = MAX( 1, J - KU ), MIN( M, J + KL )
          A( K + I, J ) = matrix( I, J )
        10    CONTINUE
      20    CONTINUE
```

Unchanged on exit.

LDA - INTEGER.
On entry, LDA specifies the first dimension of A as declared in the calling (sub) program. LDA must be at least (kl + ku + 1).
Unchanged on exit.

X - COMPLEX array of DIMENSION at least
(1 + (n - 1) * abs(INCX)) when TRANS = 'N' or 'n'
and at least
(1 + (m - 1) * abs(INCX)) otherwise.
Before entry, the incremented array X must contain the vector x.
Unchanged on exit.

INCX - INTEGER.
On entry, INCX specifies the increment for the elements of X. INCX must not be zero.
Unchanged on exit.

BETA - COMPLEX .
On entry, BETA specifies the scalar beta. When BETA is supplied as zero then Y need not be set on input.
Unchanged on exit.

Y - COMPLEX array of DIMENSION at least
(1 + (m - 1) * abs(INCY)) when TRANS = 'N' or 'n'
and at least
(1 + (n - 1) * abs(INCY)) otherwise.
Before entry, the incremented array Y must contain the vector y. On exit, Y is overwritten by the updated vector y.

INCY - INTEGER.
 On entry, INCY specifies the increment for the elements of
 Y. INCY must not be zero.
 Unchanged on exit.

***REFERENCES Dongarra, J. J., Du Croz, J., Hammarling, S., and
 Hanson, R. J. An extended set of Fortran basic linear
 algebra subprograms. ACM TOMS, Vol. 14, No. 1,
 pp. 1-17, March 1988.

***ROUTINES CALLED LSAME, XERBLA

***REVISION HISTORY (YMMDD)
 861022 DATE WRITTEN
 910605 Modified to meet SLATEC prologue standards. Only comment
 lines were modified. (BKS)
END PROLOGUE

CGBSL

```
SUBROUTINE CGBSL (ABD, LDA, N, ML, MU, IPVT, B, JOB)
***BEGIN PROLOGUE  CGBSL
***PURPOSE  Solve the complex band system  $A \cdot X = B$  or  $CTRANS(A) \cdot X = B$  using
             the factors computed by CGBCO or CGBFA.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2C2
***TYPE      COMPLEX (SGBSL-S, DGBSL-D, CGBSL-C)
***KEYWORDS  BANDED, LINEAR ALGEBRA, LINPACK, MATRIX, SOLVE
***AUTHOR    Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CGBSL solves the complex band system
 $A \cdot X = B$ or $CTRANS(A) \cdot X = B$
using the factors computed by CGBCO or CGBFA.

On Entry

ABD	COMPLEX(LDA, N)	the output from CGBCO or CGBFA.
LDA	INTEGER	the leading dimension of the array ABD .
N	INTEGER	the order of the original matrix.
ML	INTEGER	number of diagonals below the main diagonal.
MU	INTEGER	number of diagonals above the main diagonal.
IPVT	INTEGER(N)	the pivot vector from CGBCO or CGBFA.
B	COMPLEX(N)	the right hand side vector.
JOB	INTEGER	
	= 0	to solve $A \cdot X = B$,
	= nonzero	to solve $CTRANS(A) \cdot X = B$, where
		$CTRANS(A)$ is the conjugate transpose.

On Return

B the solution vector X .

Error Condition

A division by zero will occur if the input factor contains a zero on the diagonal. Technically this indicates singularity but it is often caused by improper arguments or improper setting of LDA . It will not occur if the subroutines are called correctly and if CGBCO has set RCOND .GT. 0.0 or CGBFA has set INFO .EQ. 0 .

```

      To compute  INVERSE(A) * C  where  C  is a matrix
      with  P  columns
          CALL CGBCO(ABD,LDA,N,ML,MU,IPVT,RCOND,Z)
          IF (RCOND is too small) GO TO ...
          DO 10 J = 1, P
              CALL CGBSL(ABD,LDA,N,ML,MU,IPVT,C(1,J),0)
          10 CONTINUE

***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
               Stewart, LINPACK Users' Guide, SIAM, 1979.
***ROUTINES CALLED  CAXPY, CDOTC
***REVISION HISTORY  (YYMMDD)
    780814  DATE WRITTEN
    890531  Changed all specific intrinsics to generic.  (WRB)
    890831  Modified array declarations.  (WRB)
    890831  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    900326  Removed duplicate information from DESCRIPTION section.
           (WRB)
    920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE

```

CGECO

```
SUBROUTINE CGECO (A, LDA, N, IPVT, RCOND, Z)
***BEGIN PROLOGUE  CGECO
***PURPOSE  Factor a matrix using Gaussian elimination and estimate
             the condition number of the matrix.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2C1
***TYPE      COMPLEX (SGECO-S, DGECCO-D, CGECO-C)
***KEYWORDS  CONDITION NUMBER, GENERAL MATRIX, LINEAR ALGEBRA, LINPACK,
             MATRIX FACTORIZATION
***AUTHOR  Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CGECO factors a complex matrix by Gaussian elimination and estimates the condition of the matrix.

If RCOND is not needed, CGEFA is slightly faster.
To solve $A \cdot X = B$, follow CGECO by CGESL.
To Compute $\text{INVERSE}(A) \cdot C$, follow CGECO by CGESL.
To compute $\text{DETERMINANT}(A)$, follow CGECO by CGEDI.
To compute $\text{INVERSE}(A)$, follow CGECO by CGEDI.

On Entry

A COMPLEX(LDA, N)
 the matrix to be factored.

LDA INTEGER
 the leading dimension of the array A .

N INTEGER
 the order of the matrix A .

On Return

A an upper triangular matrix and the multipliers
 which were used to obtain it.
 The factorization can be written $A = L \cdot U$ where
 L is a product of permutation and unit lower
 triangular matrices and U is upper triangular.

IPVT INTEGER(N)
 an integer vector of pivot indices.

RCOND REAL
 an estimate of the reciprocal condition of A .
 For the system $A \cdot X = B$, relative perturbations
 in A and B of size EPSILON may cause
 relative perturbations in X of size $\text{EPSILON}/\text{RCOND}$.
 If RCOND is so small that the logical expression
 $1.0 + \text{RCOND} \text{ .EQ. } 1.0$
 is true, then A may be singular to working
 precision. In particular, RCOND is zero if
 exact singularity is detected or the estimate
 underflows.

Z COMPLEX(N)

a work vector whose contents are usually unimportant.
If A is close to a singular matrix, then Z is
an approximate null vector in the sense that
 $\text{NORM}(A*Z) = \text{RCOND}*\text{NORM}(A)*\text{NORM}(Z)$.

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CAXPY, CDOTC, CGEFA, CSSCAL, SCASUM

***REVISION HISTORY (YYMMDD)

780814 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900326 Removed duplicate information from DESCRIPTION section.
(WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CGEDI

```
SUBROUTINE CGEDI (A, LDA, N, IPVT, DET, WORK, JOB)
***BEGIN PROLOGUE  CGEDI
***PURPOSE  Compute the determinant and inverse of a matrix using the
             factors computed by CGECO or CGEFA.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2C1, D3C1
***TYPE      COMPLEX (SGEDI-S, DGEDI-D, CGEDI-C)
***KEYWORDS  DETERMINANT, INVERSE, LINEAR ALGEBRA, LINPACK, MATRIX
***AUTHOR    Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CGEDI computes the determinant and inverse of a matrix using the factors computed by CGECO or CGEFA.

On Entry

A COMPLEX(LDA, N)
 the output from CGECO or CGEFA.

LDA INTEGER
 the leading dimension of the array A .

N INTEGER
 the order of the matrix A .

IPVT INTEGER(N)
 the pivot vector from CGECO or CGEFA.

WORK COMPLEX(N)
 work vector. Contents destroyed.

JOB INTEGER
 = 11 both determinant and inverse.
 = 01 inverse only.
 = 10 determinant only.

On Return

A inverse of original matrix if requested.
 Otherwise unchanged.

DET COMPLEX(2)
 determinant of original matrix if requested.
 Otherwise not referenced.
 Determinant = DET(1) * 10.0**DET(2)
 with 1.0 .LE. CABS1(DET(1)) .LT. 10.0
 or DET(1) .EQ. 0.0 .

Error Condition

A division by zero will occur if the input factor contains a zero on the diagonal and the inverse is requested. It will not occur if the subroutines are called correctly and if CGECO has set RCOND .GT. 0.0 or CGEFA has set INFO .EQ. 0 .

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
Stewart, LINPACK Users' Guide, SIAM, 1979.
***ROUTINES CALLED CAXPY, CSCAL, CSWAP
***REVISION HISTORY (YYMMDD)
780814 DATE WRITTEN
890831 Modified array declarations. (WRB)
890831 REVISION DATE from Version 3.2
891214 Prologue converted to Version 4.0 format. (BAB)
900326 Removed duplicate information from DESCRIPTION section.
(WRB)
920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

CGEEV

```
SUBROUTINE CGEEV (A, LDA, N, E, V, LDV, WORK, JOB, INFO)
***BEGIN PROLOGUE  CGEEV
***PURPOSE  Compute the eigenvalues and, optionally, the eigenvectors
             of a complex general matrix.
***LIBRARY   SLATEC
***CATEGORY  D4A4
***TYPE      COMPLEX (SGEEV-S, CGEEV-C)
***KEYWORDS  EIGENVALUES, EIGENVECTORS, GENERAL MATRIX
***AUTHOR    Kahaner, D. K., (NBS)
             Moler, C. B., (U. of New Mexico)
             Stewart, G. W., (U. of Maryland)
***DESCRIPTION
```

Abstract

CGEEV computes the eigenvalues and, optionally,
the eigenvectors of a general complex matrix.

Call Sequence Parameters-

(The values of parameters marked with * (star) will be changed
by CGEEV.)

A*	COMPLEX(LDA,N) complex nonsymmetric input matrix.
LDA	INTEGER set by the user to the leading dimension of the complex array A.
N	INTEGER set by the user to the order of the matrices A and V, and the number of elements in E.
E*	COMPLEX(N) on return from CGEEV E contains the eigenvalues of A. See also INFO below.
V*	COMPLEX(LDV,N) on return from CGEEV if the user has set JOB = 0 V is not referenced. = nonzero the N eigenvectors of A are stored in the first N columns of V. See also INFO below. (If the input matrix A is nearly degenerate, V will be badly conditioned, i.e. have nearly dependent columns.)
LDV	INTEGER set by the user to the leading dimension of the array V if JOB is also set nonzero. In that case N must be .LE. LDV. If JOB is set to zero LDV is not referenced.
WORK*	REAL(3N) temporary storage vector. Contents changed by CGEEV.
JOB	INTEGER

set by the user to
 = 0 eigenvalues only to be calculated by CGEEV.
 neither V nor LDV are referenced.
 = nonzero eigenvalues and vectors to be calculated.
 In this case A & V must be distinct arrays.
 Also, if LDA > LDV, CGEEV changes all the
 elements of A thru column N. If LDA < LDV,
 CGEEV changes all the elements of V through
 column N. If LDA = LDV only A(I,J) and V(I,
 J) for I,J = 1,...,N are changed by CGEEV.

INFO* INTEGER
 on return from CGEEV the value of INFO is
 = 0 normal return, calculation successful.
 = K if the eigenvalue iteration fails to converge,
 eigenvalues K+1 through N are correct, but
 no eigenvectors were computed even if they were
 requested (JOB nonzero).

Error Messages

No. 1	recoverable	N is greater than LDA
No. 2	recoverable	N is less than one.
No. 3	recoverable	JOB is nonzero and N is greater than LDV
No. 4	warning	LDA > LDV, elements of A other than the N by N input elements have been changed
No. 5	warning	LDA < LDV, elements of V other than the N by N output elements have been changed

***REFERENCES (NONE)

***ROUTINES CALLED CBABK2, CBAL, COMQR, COMQR2, CORTH, SCOPY, XERMSG

***REVISION HISTORY (YYMMDD)

800808 DATE WRITTEN
 890531 Changed all specific intrinsics to generic. (WRB)
 890531 REVISION DATE from Version 3.2
 891214 Prologue converted to Version 4.0 format. (BAB)
 900315 CALLs to XERROR changed to CALLs to XERMSG. (THJ)
 900326 Removed duplicate information from DESCRIPTION section.
 (WRB)

END PROLOGUE

CGEFA

```
SUBROUTINE CGEFA (A, LDA, N, IPVT, INFO)
***BEGIN PROLOGUE  CGEFA
***PURPOSE  Factor a matrix using Gaussian elimination.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2C1
***TYPE      COMPLEX (SGEFA-S, DGEFA-D, CGEFA-C)
***KEYWORDS  GENERAL MATRIX, LINEAR ALGEBRA, LINPACK,
              MATRIX FACTORIZATION
***AUTHOR   Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CGEFA factors a complex matrix by Gaussian elimination.

CGEFA is usually called by CGECO, but it can be called directly with a saving in time if RCOND is not needed.
(Time for CGECO) = (1 + 9/N)*(Time for CGEFA) .

On Entry

A COMPLEX(LDA, N)
 the matrix to be factored.

LDA INTEGER
 the leading dimension of the array A .

N INTEGER
 the order of the matrix A .

On Return

A an upper triangular matrix and the multipliers
 which were used to obtain it.
 The factorization can be written $A = L*U$ where
 L is a product of permutation and unit lower
 triangular matrices and U is upper triangular.

IPVT INTEGER(N)
 an integer vector of pivot indices.

INFO INTEGER
 = 0 normal value.
 = K if $U(K,K) \leq 0.0$. This is not an error
 condition for this subroutine, but it does
 indicate that CGESL or CGEDI will divide by zero
 if called. Use RCOND in CGECO for a reliable
 indication of singularity.

```
***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
                Stewart, LINPACK Users' Guide, SIAM, 1979.
```

```
***ROUTINES CALLED  CAXPY, CSCAL, ICAMAX
```

```
***REVISION HISTORY  (YYMMDD)
```

```
780814  DATE WRITTEN
890831  Modified array declarations.  (WRB)
890831  REVISION DATE from Version 3.2
891214  Prologue converted to Version 4.0 format.  (BAB)
900326  Removed duplicate information from DESCRIPTION section.
```

(WRB)
920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

CGEFS

```
      SUBROUTINE CGEFS (A, LDA, N, V, ITASK, IND, WORK, IWORK)
***BEGIN PROLOGUE  CGEFS
***PURPOSE  Solve a general system of linear equations.
***LIBRARY   SLATEC
***CATEGORY  D2C1
***TYPE      COMPLEX (SGEFS-S, DGEFS-D, CGEFS-C)
***KEYWORDS  COMPLEX LINEAR EQUATIONS, GENERAL MATRIX,
              GENERAL SYSTEM OF LINEAR EQUATIONS
***AUTHOR   Voorhees, E. A., (LANL)
***DESCRIPTION
```

Subroutine CGEFS solves A general NxN system of complex linear equations using LINPACK subroutines CGECO and CGESL. That is, if A is an NxN complex matrix and if X and B are complex N-vectors, then CGEFS solves the equation

$$A \cdot X = B.$$

The matrix A is first factored into upper and lower triangular matrices U and L using partial pivoting. These factors and the pivoting information are used to find the solution vector X. An approximate condition number is calculated to provide a rough estimate of the number of digits of accuracy in the computed solution.

If the equation $A \cdot X = B$ is to be solved for more than one vector B, the factoring of A does not need to be performed again and the option to only solve (ITASK .GT. 1) will be faster for the succeeding solutions. In this case, the contents of A, LDA, N and IWORK must not have been altered by the user following factorization (ITASK=1). IND will not be changed by CGEFS in this case.

Argument Description ***

A	COMPLEX(LDA,N) on entry, the doubly subscripted array with dimension (LDA,N) which contains the coefficient matrix. on return, an upper triangular matrix U and the multipliers necessary to construct a matrix L so that $A=L \cdot U$.
LDA	INTEGER the leading dimension of the array A. LDA must be greater than or equal to N. (Terminal error message IND=-1)
N	INTEGER the order of the matrix A. The first N elements of the array A are the elements of the first column of the matrix A. N must be greater than or equal to 1. (Terminal error message IND=-2)
V	COMPLEX(N) on entry, the singly subscripted array(vector) of dimension N which contains the right hand side B of a system of simultaneous linear equations $A \cdot X = B$. on return, V contains the solution vector, X.
ITASK	INTEGER

```

        if ITASK=1, the matrix A is factored and then the
            linear equation is solved.
        if ITASK .GT. 1, the equation is solved using the existing
            factored matrix A and IWORK.
        if ITASK .LT. 1, then terminal error message IND=-3 is
            printed.
IND      INTEGER
        GT.0  IND is a rough estimate of the number of digits
            of accuracy in the solution, X.
        LT.0  see error message corresponding to IND below.
WORK     COMPLEX(N)
        a singly subscripted array of dimension at least N.
IWORK    INTEGER(N)
        a singly subscripted array of dimension at least N.

```

Error Messages Printed ***

```

IND=-1  terminal    N is greater than LDA.
IND=-2  terminal    N is less than 1.
IND=-3  terminal    ITASK is less than 1.
IND=-4  terminal    The matrix A is computationally singular.
                        A solution has not been computed.
IND=-10 warning    The solution has no apparent significance.
                        The solution may be inaccurate or the matrix
                        A may be poorly scaled.

```

NOTE- The above terminal(*fatal*) error messages are designed to be handled by XERMSG in which LEVEL=1 (recoverable) and IFLAG=2 . LEVEL=0 for warning error messages from XERMSG. Unless the user provides otherwise, an error message will be printed followed by an abort.

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CGECO, CGESL, R1MACH, XERMSG

***REVISION HISTORY (YYMMDD)

```

800328  DATE WRITTEN
890531  Changed all specific intrinsics to generic.  (WRB)
890831  Modified array declarations.  (WRB)
890831  REVISION DATE from Version 3.2
891214  Prologue converted to Version 4.0 format.  (BAB)
900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
900510  Convert XERRWV calls to XERMSG calls, cvt GOTO's to
        IF-THEN-ELSE.  (RWC)
920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE

```

CGEIR

```
SUBROUTINE CGEIR (A, LDA, N, V, ITASK, IND, WORK, IWORK)
***BEGIN PROLOGUE  CGEIR
***PURPOSE  Solve a general system of linear equations.  Iterative
            refinement is used to obtain an error estimate.
***LIBRARY    SLATEC
***CATEGORY  D2C1
***TYPE       COMPLEX (SGEIR-S, CGEIR-C)
***KEYWORDS   COMPLEX LINEAR EQUATIONS, GENERAL MATRIX,
            GENERAL SYSTEM OF LINEAR EQUATIONS
***AUTHOR    Voorhees, E. A., (LANL)
***DESCRIPTION
```

Subroutine CGEIR solves a general NxN system of complex linear equations using LINPACK subroutines CGEFA and CGESL. One pass of iterative refinement is used only to obtain an estimate of the accuracy. That is, if A is an NxN complex matrix and if X and B are complex N-vectors, then CGEIR solves the equation

$$A \cdot X = B.$$

The matrix A is first factored into upper and lower triangular matrices U and L using partial pivoting. These factors and the pivoting information are used to calculate the solution, X. Then the residual vector is found and used to calculate an estimate of the relative error, IND. IND estimates the accuracy of the solution only when the input matrix and the right hand side are represented exactly in the computer and does not take into account any errors in the input data.

If the equation $A \cdot X = B$ is to be solved for more than one vector B, the factoring of A does not need to be performed again and the option to only solve (ITASK .GT. 1) will be faster for the succeeding solutions. In this case, the contents of A, LDA, N, WORK, and IWORK must not have been altered by the user following factorization (ITASK=1). IND will not be changed by CGEIR in this case.

Argument Description ***

A	COMPLEX(LDA,N) the doubly subscripted array with dimension (LDA,N) which contains the coefficient matrix. A is not altered by the routine.
LDA	INTEGER the leading dimension of the array A. LDA must be greater than or equal to N. (Terminal error message IND=-1)
N	INTEGER the order of the matrix A. The first N elements of the array A are the elements of the first column of matrix A. N must be greater than or equal to 1. (Terminal error message IND=-2)
V	COMPLEX(N) on entry, the singly subscripted array(vector) of dimension N which contains the right hand side B of a

```

        system of simultaneous linear equations  $A \cdot X = B$ .
        on return, V contains the solution vector, X .
ITASK  INTEGER
        if ITASK=1, the matrix A is factored and then the
           linear equation is solved.
        if ITASK .GT. 1, the equation is solved using the existing
           factored matrix A (stored in work).
        if ITASK .LT. 1, then terminal error message IND=-3 is
           printed.
IND     INTEGER
        GT.0  IND is a rough estimate of the number of digits
              of accuracy in the solution, X.  IND=75 means
              that the solution vector X is zero.
        LT.0  see error message corresponding to IND below.
WORK    COMPLEX(N*(N+1))
        a singly subscripted array of dimension at least N*(N+1).
IWORK   INTEGER(N)
        a singly subscripted array of dimension at least N.

```

Error Messages Printed ***

```

IND=-1  terminal    N is greater than LDA.
IND=-2  terminal    N is less than one.
IND=-3  terminal    ITASK is less than one.
IND=-4  terminal    The matrix A is computationally singular.
                   A solution has not been computed.
IND=-10 warning    The solution has no apparent significance.
                   The solution may be inaccurate or the matrix
                   A may be poorly scaled.

```

NOTE- The above terminal(*fatal*) error messages are designed to be handled by XERMSG in which LEVEL=1 (recoverable) and IFLAG=2 . LEVEL=0 for warning error messages from XERMSG. Unless the user provides otherwise, an error message will be printed followed by an abort.

```

***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
                Stewart, LINPACK Users' Guide, SIAM, 1979.
***ROUTINES CALLED  CCOPY, CDCDOT, CGEFA, CGESL, R1MACH, SCASUM, XERMSG
***REVISION HISTORY  (YYMMDD)
800502  DATE WRITTEN
890531  Changed all specific intrinsics to generic.  (WRB)
890831  Modified array declarations.  (WRB)
890831  REVISION DATE from Version 3.2
891214  Prologue converted to Version 4.0 format.  (BAB)
900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
900510  Convert XERRWV calls to XERMSG calls, cvt GOTO's to
        IF-THEN-ELSE.  (RWC)
920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE

```

CGEMM

```
      SUBROUTINE CGEMM (TRANSA, TRANSB, M, N, K, ALPHA, A, LDA, B, LDB,  
$      BETA, C, LDC)  
***BEGIN PROLOGUE  CGEMM  
***PURPOSE  Multiply a complex general matrix by a complex general  
            matrix.  
***LIBRARY    SLATEC (BLAS)  
***CATEGORY   D1B6  
***TYPE       COMPLEX (SGEMM-S, DGEMM-D, CGEMM-C)  
***KEYWORDS   LEVEL 3 BLAS, LINEAR ALGEBRA  
***AUTHOR     Dongarra, J., (ANL)  
              Duff, I., (AERE)  
              Du Croz, J., (NAG)  
              Hammarling, S. (NAG)  
***DESCRIPTION
```

CGEMM performs one of the matrix-matrix operations

$$C := \alpha * \text{op}(A) * \text{op}(B) + \beta * C,$$

where $\text{op}(X)$ is one of

$$\text{op}(X) = X \quad \text{or} \quad \text{op}(X) = X' \quad \text{or} \quad \text{op}(X) = \text{conjg}(X'),$$

α and β are scalars, and A , B and C are matrices, with $\text{op}(A)$ an m by k matrix, $\text{op}(B)$ a k by n matrix and C an m by n matrix.

Parameters
=====

TRANSA - CHARACTER*1.

On entry, TRANSA specifies the form of $\text{op}(A)$ to be used in the matrix multiplication as follows:

TRANSA = 'N' or 'n', $\text{op}(A) = A$.

TRANSA = 'T' or 't', $\text{op}(A) = A'$.

TRANSA = 'C' or 'c', $\text{op}(A) = \text{conjg}(A')$.

Unchanged on exit.

TRANSB - CHARACTER*1.

On entry, TRANSB specifies the form of $\text{op}(B)$ to be used in the matrix multiplication as follows:

TRANSB = 'N' or 'n', $\text{op}(B) = B$.

TRANSB = 'T' or 't', $\text{op}(B) = B'$.

TRANSB = 'C' or 'c', $\text{op}(B) = \text{conjg}(B')$.

Unchanged on exit.

M - INTEGER.

On entry, M specifies the number of rows of the matrix $\text{op}(A)$ and of the matrix C . M must be at least zero.

Unchanged on exit.

N - INTEGER.
On entry, N specifies the number of columns of the matrix $\text{op}(B)$ and the number of columns of the matrix C. N must be at least zero.
Unchanged on exit.

K - INTEGER.
On entry, K specifies the number of columns of the matrix $\text{op}(A)$ and the number of rows of the matrix $\text{op}(B)$. K must be at least zero.
Unchanged on exit.

ALPHA - COMPLEX .
On entry, ALPHA specifies the scalar alpha.
Unchanged on exit.

A - COMPLEX array of DIMENSION (LDA, ka), where ka is k when $\text{TRANSA} = 'N'$ or $'n'$, and is m otherwise.
Before entry with $\text{TRANSA} = 'N'$ or $'n'$, the leading m by k part of the array A must contain the matrix A, otherwise the leading k by m part of the array A must contain the matrix A.
Unchanged on exit.

LDA - INTEGER.
On entry, LDA specifies the first dimension of A as declared in the calling (sub) program. When $\text{TRANSA} = 'N'$ or $'n'$ then LDA must be at least $\max(1, m)$, otherwise LDA must be at least $\max(1, k)$.
Unchanged on exit.

B - COMPLEX array of DIMENSION (LDB, kb), where kb is n when $\text{TRANSB} = 'N'$ or $'n'$, and is k otherwise.
Before entry with $\text{TRANSB} = 'N'$ or $'n'$, the leading k by n part of the array B must contain the matrix B, otherwise the leading n by k part of the array B must contain the matrix B.
Unchanged on exit.

LDB - INTEGER.
On entry, LDB specifies the first dimension of B as declared in the calling (sub) program. When $\text{TRANSB} = 'N'$ or $'n'$ then LDB must be at least $\max(1, k)$, otherwise LDB must be at least $\max(1, n)$.
Unchanged on exit.

BETA - COMPLEX .
On entry, BETA specifies the scalar beta. When BETA is supplied as zero then C need not be set on input.
Unchanged on exit.

C - COMPLEX array of DIMENSION (LDC, n).
Before entry, the leading m by n part of the array C must contain the matrix C, except when beta is zero, in which case C need not be set on entry.
On exit, the array C is overwritten by the m by n matrix $(\alpha * \text{op}(A) * \text{op}(B) + \beta * C)$.

LDC - INTEGER.
 On entry, LDC specifies the first dimension of C as declared
 in the calling (sub) program. LDC must be at least
 max(1, m).
 Unchanged on exit.

***REFERENCES Dongarra, J., Du Croz, J., Duff, I., and Hammarling, S.
 A set of level 3 basic linear algebra subprograms.
 ACM TOMS, Vol. 16, No. 1, pp. 1-17, March 1990.

***ROUTINES CALLED LSAME, XERBLA

***REVISION HISTORY (YYMMDD)
 890208 DATE WRITTEN
 910605 Modified to meet SLATEC prologue standards. Only comment
 lines were modified. (BKS)
 END PROLOGUE

CGEMV

```
      SUBROUTINE CGEMV (TRANS, M, N, ALPHA, A, LDA, X, INCX, BETA, Y,  
$      INCY)  
***BEGIN PROLOGUE  CGEMV  
***PURPOSE  Multiply a complex vector by a complex general matrix.  
***LIBRARY   SLATEC (BLAS)  
***CATEGORY  D1B4  
***TYPE      COMPLEX (SGEMV-S, DGEMV-D, CGEMV-C)  
***KEYWORDS  LEVEL 2 BLAS, LINEAR ALGEBRA  
***AUTHOR   Dongarra, J. J., (ANL)  
            Du Croz, J., (NAG)  
            Hammarling, S., (NAG)  
            Hanson, R. J., (SNLA)  
***DESCRIPTION
```

CGEMV performs one of the matrix-vector operations

$$y := \alpha A x + \beta y, \quad \text{or} \quad y := \alpha A' x + \beta y, \quad \text{or}$$
$$y := \alpha \text{conjg}(A') x + \beta y,$$

where α and β are scalars, x and y are vectors and A is an m by n matrix.

Parameters
=====

TRANS - CHARACTER*1.
On entry, TRANS specifies the operation to be performed as follows:

$$\begin{aligned} \text{TRANS} = \text{'N' or 'n'} & \quad y := \alpha A x + \beta y. \\ \text{TRANS} = \text{'T' or 't'} & \quad y := \alpha A' x + \beta y. \\ \text{TRANS} = \text{'C' or 'c'} & \quad y := \alpha \text{conjg}(A') x + \beta y. \end{aligned}$$

Unchanged on exit.

M - INTEGER.
On entry, M specifies the number of rows of the matrix A.
M must be at least zero.
Unchanged on exit.

N - INTEGER.
On entry, N specifies the number of columns of the matrix A.
N must be at least zero.
Unchanged on exit.

ALPHA - COMPLEX .
On entry, ALPHA specifies the scalar α .
Unchanged on exit.

A - COMPLEX array of DIMENSION (LDA, n).
Before entry, the leading m by n part of the array A must contain the matrix of coefficients.
Unchanged on exit.

LDA - INTEGER.
On entry, LDA specifies the first dimension of A as declared in the calling (sub) program. LDA must be at least $\max(1, m)$.
Unchanged on exit.

X - COMPLEX array of DIMENSION at least $(1 + (n - 1) * \text{abs}(\text{INCX}))$ when TRANS = 'N' or 'n' and at least $(1 + (m - 1) * \text{abs}(\text{INCX}))$ otherwise.
Before entry, the incremented array X must contain the vector x.
Unchanged on exit.

INCX - INTEGER.
On entry, INCX specifies the increment for the elements of X. INCX must not be zero.
Unchanged on exit.

BETA - COMPLEX .
On entry, BETA specifies the scalar beta. When BETA is supplied as zero then Y need not be set on input.
Unchanged on exit.

Y - COMPLEX array of DIMENSION at least $(1 + (m - 1) * \text{abs}(\text{INCY}))$ when TRANS = 'N' or 'n' and at least $(1 + (n - 1) * \text{abs}(\text{INCY}))$ otherwise.
Before entry with BETA non-zero, the incremented array Y must contain the vector y. On exit, Y is overwritten by the updated vector y.

INCY - INTEGER.
On entry, INCY specifies the increment for the elements of Y. INCY must not be zero.
Unchanged on exit.

***REFERENCES Dongarra, J. J., Du Croz, J., Hammarling, S., and Hanson, R. J. An extended set of Fortran basic linear algebra subprograms. ACM TOMS, Vol. 14, No. 1, pp. 1-17, March 1988.

***ROUTINES CALLED LSAME, XERBLA

***REVISION HISTORY (YYMMDD)
861022 DATE WRITTEN
910605 Modified to meet SLATEC prologue standards. Only comment lines were modified. (BKS)
END PROLOGUE

CGERC

```
      SUBROUTINE CGERC (M, N, ALPHA, X, INCX, Y, INCY, A, LDA)
***BEGIN PROLOGUE  CGERC
***PURPOSE  Perform conjugated rank 1 update of a complex general
            matrix.
***LIBRARY   SLATEC (BLAS)
***CATEGORY  D1B4
***TYPE      COMPLEX (SGERC-S, DGERC-D, CGERC-C)
***KEYWORDS  LEVEL 2 BLAS, LINEAR ALGEBRA
***AUTHOR   Dongarra, J. J., (ANL)
            Du Croz, J., (NAG)
            Hammarling, S., (NAG)
            Hanson, R. J., (SNLA)
***DESCRIPTION
```

CGERC performs the rank 1 operation

$$A := \alpha * x * \text{conjg}(y') + A,$$

where alpha is a scalar, x is an m element vector, y is an n element vector and A is an m by n matrix.

Parameters
=====

M - INTEGER.
 On entry, M specifies the number of rows of the matrix A.
 M must be at least zero.
 Unchanged on exit.

N - INTEGER.
 On entry, N specifies the number of columns of the matrix A.
 N must be at least zero.
 Unchanged on exit.

ALPHA - COMPLEX .
 On entry, ALPHA specifies the scalar alpha.
 Unchanged on exit.

X - COMPLEX array of dimension at least
 (1 + (m - 1) * abs(INCX)).
 Before entry, the incremented array X must contain the m
 element vector x.
 Unchanged on exit.

INCX - INTEGER.
 On entry, INCX specifies the increment for the elements of
 X. INCX must not be zero.
 Unchanged on exit.

Y - COMPLEX array of dimension at least
 (1 + (n - 1) * abs(INCY)).
 Before entry, the incremented array Y must contain the n
 element vector y.
 Unchanged on exit.

INCY - INTEGER.

On entry, INCY specifies the increment for the elements of Y. INCY must not be zero.
Unchanged on exit.

A - COMPLEX array of DIMENSION (LDA, n).
Before entry, the leading m by n part of the array A must contain the matrix of coefficients. On exit, A is overwritten by the updated matrix.

LDA - INTEGER.
On entry, LDA specifies the first dimension of A as declared in the calling (sub) program. LDA must be at least max(1, m).
Unchanged on exit.

***REFERENCES Dongarra, J. J., Du Croz, J., Hammarling, S., and Hanson, R. J. An extended set of Fortran basic linear algebra subprograms. ACM TOMS, Vol. 14, No. 1, pp. 1-17, March 1988.

***ROUTINES CALLED XERBLA

***REVISION HISTORY (YYMMDD)

861022 DATE WRITTEN

910605 Modified to meet SLATEC prologue standards. Only comment lines were modified. (BKS)

END PROLOGUE

CGERU

```
SUBROUTINE CGERU (M, N, ALPHA, X, INCX, Y, INCY, A, LDA)
***BEGIN PROLOGUE  CGERU
***PURPOSE  Perform unconjugated rank 1 update of a complex general
            matrix.
***LIBRARY   SLATEC (BLAS)
***CATEGORY  D1B4
***TYPE      COMPLEX (SGERU-S, DGERU-D, CGERU-C)
***KEYWORDS  LEVEL 2 BLAS, LINEAR ALGEBRA
***AUTHOR    Dongarra, J. J., (ANL)
            Du Croz, J., (NAG)
            Hammarling, S., (NAG)
            Hanson, R. J., (SNLA)
***DESCRIPTION
```

CGERU performs the rank 1 operation

$$A := \alpha x y^H + A,$$

where α is a scalar, x is an m element vector, y is an n element vector and A is an m by n matrix.

Parameters
=====

M - INTEGER.
On entry, M specifies the number of rows of the matrix A.
M must be at least zero.
Unchanged on exit.

N - INTEGER.
On entry, N specifies the number of columns of the matrix A.
N must be at least zero.
Unchanged on exit.

ALPHA - COMPLEX .
On entry, ALPHA specifies the scalar α .
Unchanged on exit.

X - COMPLEX array of dimension at least
(1 + (m - 1) * abs(INCX)).
Before entry, the incremented array X must contain the m
element vector x .
Unchanged on exit.

INCX - INTEGER.
On entry, INCX specifies the increment for the elements of
X. INCX must not be zero.
Unchanged on exit.

Y - COMPLEX array of dimension at least
(1 + (n - 1) * abs(INCY)).
Before entry, the incremented array Y must contain the n
element vector y .
Unchanged on exit.

INCY - INTEGER.

On entry, INCY specifies the increment for the elements of Y. INCY must not be zero.
Unchanged on exit.

A - COMPLEX array of DIMENSION (LDA, n).
Before entry, the leading m by n part of the array A must contain the matrix of coefficients. On exit, A is overwritten by the updated matrix.

LDA - INTEGER.
On entry, LDA specifies the first dimension of A as declared in the calling (sub) program. LDA must be at least max(1, m).
Unchanged on exit.

***REFERENCES Dongarra, J. J., Du Croz, J., Hammarling, S., and Hanson, R. J. An extended set of Fortran basic linear algebra subprograms. ACM TOMS, Vol. 14, No. 1, pp. 1-17, March 1988.

***ROUTINES CALLED XERBLA

***REVISION HISTORY (YYMMDD)

861022 DATE WRITTEN

910605 Modified to meet SLATEC prologue standards. Only comment lines were modified. (BKS)

END PROLOGUE

CGESL

```
SUBROUTINE CGESL (A, LDA, N, IPVT, B, JOB)
***BEGIN PROLOGUE  CGESL
***PURPOSE  Solve the complex system A*X=B or CTRANS(A)*X=B using the
             factors computed by CGECO or CGEFA.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2C1
***TYPE      COMPLEX (SGESL-S, DGESL-D, CGESL-C)
***KEYWORDS  LINEAR ALGEBRA, LINPACK, MATRIX, SOLVE
***AUTHOR    Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CGESL solves the complex system
 $A * X = B$ or $CTRANS(A) * X = B$
using the factors computed by CGECO or CGEFA.

On Entry

A COMPLEX(LDA, N)
 the output from CGECO or CGEFA.

LDA INTEGER
 the leading dimension of the array A .

N INTEGER
 the order of the matrix A .

IPVT INTEGER(N)
 the pivot vector from CGECO or CGEFA.

B COMPLEX(N)
 the right hand side vector.

JOB INTEGER
 = 0 to solve $A * X = B$,
 = nonzero to solve $CTRANS(A) * X = B$ where
 CTRANS(A) is the conjugate transpose.

On Return

B the solution vector X .

Error Condition

A division by zero will occur if the input factor contains a zero on the diagonal. Technically this indicates singularity but it is often caused by improper arguments or improper setting of LDA . It will not occur if the subroutines are called correctly and if CGECO has set RCOND .GT. 0.0 or CGEFA has set INFO .EQ. 0 .

To compute $INVERSE(A) * C$ where C is a matrix with P columns

```
CALL CGECO(A,LDA,N,IPVT,RCOND,Z)
IF (RCOND is too small) GO TO ...
DO 10 J = 1, P
  CALL CGESL(A,LDA,N,IPVT,C(1,J),0)
```

10 CONTINUE

```
***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
                Stewart, LINPACK Users' Guide, SIAM, 1979.
***ROUTINES CALLED  CAXPY, CDOTC
***REVISION HISTORY  (YYMMDD)
  780814  DATE WRITTEN
  890831  Modified array declarations.  (WRB)
  890831  REVISION DATE from Version 3.2
  891214  Prologue converted to Version 4.0 format.  (BAB)
  900326  Removed duplicate information from DESCRIPTION section.
          (WRB)
  920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

CGTSL

```
SUBROUTINE CGTSL (N, C, D, E, B, INFO)
***BEGIN PROLOGUE  CGTSL
***PURPOSE  Solve a tridiagonal linear system.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2C2A
***TYPE      COMPLEX (SGTSL-S, DGTSL-D, CGTSL-C)
***KEYWORDS  LINEAR ALGEBRA, LINPACK, MATRIX, SOLVE, TRIDIAGONAL
***AUTHOR   Dongarra, J., (ANL)
***DESCRIPTION
```

CGTSL given a general tridiagonal matrix and a right hand side will find the solution.

On Entry

N INTEGER
 is the order of the tridiagonal matrix.

C COMPLEX(N)
 is the subdiagonal of the tridiagonal matrix.
 C(2) through C(N) should contain the subdiagonal.
 On output C is destroyed.

D COMPLEX(N)
 is the diagonal of the tridiagonal matrix.
 On output D is destroyed.

E COMPLEX(N)
 is the superdiagonal of the tridiagonal matrix.
 E(1) through E(N-1) should contain the superdiagonal.
 On output E is destroyed.

B COMPLEX(N)
 is the right hand side vector.

On Return

B is the solution vector.

INFO INTEGER
 = 0 normal value.
 = K if the K-th element of the diagonal becomes
 exactly zero. The subroutine returns when
 this is detected.

```
***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
                Stewart, LINPACK Users' Guide, SIAM, 1979.
```

```
***ROUTINES CALLED  (NONE)
```

```
***REVISION HISTORY  (YYMMDD)
```

```
780814  DATE WRITTEN
890831  Modified array declarations.  (WRB)
890831  REVISION DATE from Version 3.2
891214  Prologue converted to Version 4.0 format.  (BAB)
900326  Removed duplicate information from DESCRIPTION section.
        (WRB)
```

920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

CH

```
      SUBROUTINE CH (NM, N, AR, AI, W, MATZ, ZR, ZI, FV1, FV2, FM1,  
+      IERR)  
***BEGIN PROLOGUE  CH  
***PURPOSE  Compute the eigenvalues and, optionally, the eigenvectors  
             of a complex Hermitian matrix.  
***LIBRARY   SLATEC (EISPACK)  
***CATEGORY  D4A3  
***TYPE      COMPLEX (RS-S, CH-C)  
***KEYWORDS  EIGENVALUES, EIGENVECTORS, EISPACK  
***AUTHOR   Smith, B. T., et al.  
***DESCRIPTION
```

This subroutine calls the recommended sequence of subroutines from the eigensystem subroutine package (EISPACK) to find the eigenvalues and eigenvectors (if desired) of a COMPLEX HERMITIAN matrix.

On INPUT

NM must be set to the row dimension of the two-dimensional array parameters, AR, AI, ZR and ZI, as declared in the calling program dimension statement. NM is an INTEGER variable.

N is the order of the matrix A=(AR,AI). N is an INTEGER variable. N must be less than or equal to NM.

AR and AI contain the real and imaginary parts, respectively, of the complex Hermitian matrix. AR and AI are two-dimensional REAL arrays, dimensioned AR(NM,N) and AI(NM,N).

MATZ is an INTEGER variable set equal to zero if only eigenvalues are desired. Otherwise, it is set to any non-zero integer for both eigenvalues and eigenvectors.

On OUTPUT

W contains the eigenvalues in ascending order.
W is a one-dimensional REAL array, dimensioned W(N).

ZR and ZI contain the real and imaginary parts, respectively, of the eigenvectors if MATZ is not zero. ZR and ZI are two-dimensional REAL arrays, dimensioned ZR(NM,N) and ZI(NM,N).

IERR is an INTEGER flag set to
Zero for normal return,
10*N if N is greater than NM,
J if the J-th eigenvalue has not been
 determined after a total of 30 iterations.
The eigenvalues should be correct for indices
1, 2, ..., IERR-1, but no eigenvectors are
computed.

FV1 and FV2 are one-dimensional REAL arrays used for

temporary storage, dimensioned FV1(N) and FV2(N).

FM1 is a two-dimensional REAL array used for temporary storage, dimensioned FM1(2,N).

Questions and comments should be directed to B. S. Garbow,
APPLIED MATHEMATICS DIVISION, ARGONNE NATIONAL LABORATORY

***REFERENCES B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow,
Y. Ikebe, V. C. Klema and C. B. Moler, Matrix Eigen-
system Routines - EISPACK Guide, Springer-Verlag,
1976.

***ROUTINES CALLED HTRIBK, HTRIDI, TQL2, TQLRAT

***REVISION HISTORY (YYMMDD)

760101 DATE WRITTEN

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CHBMV

```
      SUBROUTINE CHBMV (UPLO, N, K, ALPHA, A, LDA, X, INCX, BETA, Y,  
$      INCY)  
***BEGIN PROLOGUE  CHBMV  
***PURPOSE  Multiply a complex vector by a complex Hermitian band  
            matrix.  
***LIBRARY    SLATEC (BLAS)  
***CATEGORY   D1B4  
***TYPE       COMPLEX (SHBMV-S, DHBMV-D, CHBMV-C)  
***KEYWORDS   LEVEL 2 BLAS, LINEAR ALGEBRA  
***AUTHOR     Dongarra, J. J., (ANL)  
              Du Croz, J., (NAG)  
              Hammarling, S., (NAG)  
              Hanson, R. J., (SNLA)  
***DESCRIPTION
```

CHBMV performs the matrix-vector operation

$$y := \alpha A x + \beta y,$$

where α and β are scalars, x and y are n element vectors and A is an n by n hermitian band matrix, with k super-diagonals.

Parameters

=====

UPLO - CHARACTER*1.
On entry, UPLO specifies whether the upper or lower triangular part of the band matrix A is being supplied as follows:

UPLO = 'U' or 'u'	The upper triangular part of A is being supplied.
UPLO = 'L' or 'l'	The lower triangular part of A is being supplied.

Unchanged on exit.

N - INTEGER.
On entry, N specifies the order of the matrix A .
 N must be at least zero.
Unchanged on exit.

K - INTEGER.
On entry, K specifies the number of super-diagonals of the matrix A . K must satisfy $0 \leq K$.
Unchanged on exit.

ALPHA - COMPLEX .
On entry, ALPHA specifies the scalar α .
Unchanged on exit.

A - COMPLEX array of DIMENSION (LDA, n).
Before entry with UPLO = 'U' or 'u', the leading ($k + 1$) by n part of the array A must contain the upper triangular band part of the hermitian matrix, supplied column by

column, with the leading diagonal of the matrix in row ($k + 1$) of the array, the first super-diagonal starting at position 2 in row k , and so on. The top left k by k triangle of the array A is not referenced.

The following program segment will transfer the upper triangular part of a hermitian band matrix from conventional full matrix storage to band storage:

```

      DO 20, J = 1, N
        M = K + 1 - J
        DO 10, I = MAX( 1, J - K ), J
          A( M + I, J ) = matrix( I, J )
10      CONTINUE
20 CONTINUE

```

Before entry with UPLO = 'L' or 'l', the leading ($k + 1$) by n part of the array A must contain the lower triangular band part of the hermitian matrix, supplied column by column, with the leading diagonal of the matrix in row 1 of the array, the first sub-diagonal starting at position 1 in row 2, and so on. The bottom right k by k triangle of the array A is not referenced.

The following program segment will transfer the lower triangular part of a hermitian band matrix from conventional full matrix storage to band storage:

```

      DO 20, J = 1, N
        M = 1 - J
        DO 10, I = J, MIN( N, J + K )
          A( M + I, J ) = matrix( I, J )
10      CONTINUE
20 CONTINUE

```

Note that the imaginary parts of the diagonal elements need not be set and are assumed to be zero.
Unchanged on exit.

- LDA - INTEGER.
On entry, LDA specifies the first dimension of A as declared in the calling (sub) program. LDA must be at least ($k + 1$).
Unchanged on exit.
- X - COMPLEX array of DIMENSION at least
($1 + (n - 1) * \text{abs}(\text{INCX})$).
Before entry, the incremented array X must contain the vector x .
Unchanged on exit.
- INCX - INTEGER.
On entry, INCX specifies the increment for the elements of X . INCX must not be zero.
Unchanged on exit.
- BETA - COMPLEX .
On entry, BETA specifies the scalar beta.
Unchanged on exit.
- Y - COMPLEX array of DIMENSION at least
($1 + (n - 1) * \text{abs}(\text{INCY})$).

Before entry, the incremented array Y must contain the vector y. On exit, Y is overwritten by the updated vector y.

INCY - INTEGER.

On entry, INCY specifies the increment for the elements of Y. INCY must not be zero.

Unchanged on exit.

***REFERENCES Dongarra, J. J., Du Croz, J., Hammarling, S., and Hanson, R. J. An extended set of Fortran basic linear algebra subprograms. ACM TOMS, Vol. 14, No. 1, pp. 1-17, March 1988.

***ROUTINES CALLED LSAME, XERBLA

***REVISION HISTORY (YYMMDD)

861022 DATE WRITTEN

910605 Modified to meet SLATEC prologue standards. Only comment lines were modified. (BKS)

END PROLOGUE

CHEMM

```
      SUBROUTINE CHEMM (SIDE, UPLO, M, N, ALPHA, A, LDA, B, LDB, BETA,  
$      C, LDC)  
***BEGIN PROLOGUE  CHEMM  
***PURPOSE  Multiply a complex general matrix by a complex Hermitian  
            matrix.  
***LIBRARY   SLATEC (BLAS)  
***CATEGORY  D1B6  
***TYPE      COMPLEX (SHEMM-S, DHEMM-D, CHEMM-C)  
***KEYWORDS  LEVEL 3 BLAS, LINEAR ALGEBRA  
***AUTHOR    Dongarra, J., (ANL)  
            Duff, I., (AERE)  
            Du Croz, J., (NAG)  
            Hammarling, S. (NAG)  
***DESCRIPTION
```

CHEMM performs one of the matrix-matrix operations

$$C := \alpha A^*B + \beta C,$$

or

$$C := \alpha B^*A + \beta C,$$

where alpha and beta are scalars, A is an hermitian matrix and B and C are m by n matrices.

Parameters
=====

SIDE - CHARACTER*1.
On entry, SIDE specifies whether the hermitian matrix A appears on the left or right in the operation as follows:

SIDE = 'L' or 'l' $C := \alpha A^*B + \beta C,$

SIDE = 'R' or 'r' $C := \alpha B^*A + \beta C,$

Unchanged on exit.

UPLO - CHARACTER*1.
On entry, UPLO specifies whether the upper or lower triangular part of the hermitian matrix A is to be referenced as follows:

UPLO = 'U' or 'u' Only the upper triangular part of the hermitian matrix is to be referenced.

UPLO = 'L' or 'l' Only the lower triangular part of the hermitian matrix is to be referenced.

Unchanged on exit.

M - INTEGER.
On entry, M specifies the number of rows of the matrix C. M must be at least zero.
Unchanged on exit.

- N - INTEGER.
On entry, N specifies the number of columns of the matrix C.
N must be at least zero.
Unchanged on exit.
- ALPHA - COMPLEX .
On entry, ALPHA specifies the scalar alpha.
Unchanged on exit.
- A - COMPLEX array of DIMENSION (LDA, ka), where ka is m when SIDE = 'L' or 'l' and is n otherwise.
Before entry with SIDE = 'L' or 'l', the m by m part of the array A must contain the hermitian matrix, such that when UPLO = 'U' or 'u', the leading m by m upper triangular part of the array A must contain the upper triangular part of the hermitian matrix and the strictly lower triangular part of A is not referenced, and when UPLO = 'L' or 'l', the leading m by m lower triangular part of the array A must contain the lower triangular part of the hermitian matrix and the strictly upper triangular part of A is not referenced.
Before entry with SIDE = 'R' or 'r', the n by n part of the array A must contain the hermitian matrix, such that when UPLO = 'U' or 'u', the leading n by n upper triangular part of the array A must contain the upper triangular part of the hermitian matrix and the strictly lower triangular part of A is not referenced, and when UPLO = 'L' or 'l', the leading n by n lower triangular part of the array A must contain the lower triangular part of the hermitian matrix and the strictly upper triangular part of A is not referenced.
Note that the imaginary parts of the diagonal elements need not be set, they are assumed to be zero.
Unchanged on exit.
- LDA - INTEGER.
On entry, LDA specifies the first dimension of A as declared in the calling (sub) program. When SIDE = 'L' or 'l' then LDA must be at least max(1, m), otherwise LDA must be at least max(1, n).
Unchanged on exit.
- B - COMPLEX array of DIMENSION (LDB, n).
Before entry, the leading m by n part of the array B must contain the matrix B.
Unchanged on exit.
- LDB - INTEGER.
On entry, LDB specifies the first dimension of B as declared in the calling (sub) program. LDB must be at least max(1, m).
Unchanged on exit.
- BETA - COMPLEX .
On entry, BETA specifies the scalar beta. When BETA is supplied as zero then C need not be set on input.
Unchanged on exit.
- C - COMPLEX array of DIMENSION (LDC, n).

Before entry, the leading m by n part of the array C must contain the matrix C , except when β is zero, in which case C need not be set on entry. On exit, the array C is overwritten by the m by n updated matrix.

LDC - INTEGER.

On entry, LDC specifies the first dimension of C as declared in the calling (sub) program. LDC must be at least $\max(1, m)$. Unchanged on exit.

***REFERENCES Dongarra, J., Du Croz, J., Duff, I., and Hammarling, S.
A set of level 3 basic linear algebra subprograms.
ACM TOMS, Vol. 16, No. 1, pp. 1-17, March 1990.

***ROUTINES CALLED LSAME, XERBLA

***REVISION HISTORY (YYMMDD)

890208 DATE WRITTEN

910605 Modified to meet SLATEC prologue standards. Only comment lines were modified. (BKS)

END PROLOGUE

CHEMV

```
SUBROUTINE CHEMV (UPLO, N, ALPHA, A, LDA, X, INCX, BETA, Y, INCY)
***BEGIN PROLOGUE  CHEMV
***PURPOSE  Multiply a complex vector by a complex Hermitian matrix.
***LIBRARY   SLATEC (BLAS)
***CATEGORY  D1B4
***TYPE      COMPLEX (SHEMV-S, DHEMV-D, CHEMV-C)
***KEYWORDS  LEVEL 2 BLAS, LINEAR ALGEBRA
***AUTHOR   Dongarra, J. J., (ANL)
            Du Croz, J., (NAG)
            Hammarling, S., (NAG)
            Hanson, R. J., (SNLA)
***DESCRIPTION
```

CHEMV performs the matrix-vector operation

$$y := \alpha A x + \beta y,$$

where α and β are scalars, x and y are n element vectors and A is an n by n hermitian matrix.

Parameters
=====

UPLO - CHARACTER*1.
On entry, UPLO specifies whether the upper or lower triangular part of the array A is to be referenced as follows:

UPLO = 'U' or 'u' Only the upper triangular part of A is to be referenced.

UPLO = 'L' or 'l' Only the lower triangular part of A is to be referenced.

Unchanged on exit.

N - INTEGER.
On entry, N specifies the order of the matrix A.
N must be at least zero.
Unchanged on exit.

ALPHA - COMPLEX .
On entry, ALPHA specifies the scalar α .
Unchanged on exit.

A - COMPLEX array of DIMENSION (LDA, n).
Before entry with UPLO = 'U' or 'u', the leading n by n upper triangular part of the array A must contain the upper triangular part of the hermitian matrix and the strictly lower triangular part of A is not referenced.
Before entry with UPLO = 'L' or 'l', the leading n by n lower triangular part of the array A must contain the lower triangular part of the hermitian matrix and the strictly upper triangular part of A is not referenced.
Note that the imaginary parts of the diagonal elements need not be set and are assumed to be zero.

Unchanged on exit.

LDA - INTEGER.
On entry, LDA specifies the first dimension of A as declared in the calling (sub) program. LDA must be at least $\max(1, n)$.
Unchanged on exit.

X - COMPLEX array of dimension at least $(1 + (n - 1) * \text{abs}(\text{INCX}))$.
Before entry, the incremented array X must contain the n element vector x.
Unchanged on exit.

INCX - INTEGER.
On entry, INCX specifies the increment for the elements of X. INCX must not be zero.
Unchanged on exit.

BETA - COMPLEX .
On entry, BETA specifies the scalar beta. When BETA is supplied as zero then Y need not be set on input.
Unchanged on exit.

Y - COMPLEX array of dimension at least $(1 + (n - 1) * \text{abs}(\text{INCY}))$.
Before entry, the incremented array Y must contain the n element vector y. On exit, Y is overwritten by the updated vector y.

INCY - INTEGER.
On entry, INCY specifies the increment for the elements of Y. INCY must not be zero.
Unchanged on exit.

***REFERENCES Dongarra, J. J., Du Croz, J., Hammarling, S., and Hanson, R. J. An extended set of Fortran basic linear algebra subprograms. ACM TOMS, Vol. 14, No. 1, pp. 1-17, March 1988.

***ROUTINES CALLED LSAME, XERBLA

***REVISION HISTORY (YYMMDD)

861022 DATE WRITTEN

910605 Modified to meet SLATEC prologue standards. Only comment lines were modified. (BKS)

END PROLOGUE

CHER

```
SUBROUTINE CHER (UPLO, N, ALPHA, X, INCX, A, LDA)
***BEGIN PROLOGUE  CHER
***PURPOSE  Perform Hermitian rank 1 update of a complex Hermitian
            matrix.
***LIBRARY    SLATEC (BLAS)
***CATEGORY   D1B4
***TYPE       COMPLEX (SHER-S, DHER-D, CHER-C)
***KEYWORDS   LEVEL 2 BLAS, LINEAR ALGEBRA
***AUTHOR     Dongarra, J. J., (ANL)
            Du Croz, J., (NAG)
            Hammarling, S., (NAG)
            Hanson, R. J., (SNLA)
***DESCRIPTION
```

CHER performs the hermitian rank 1 operation

$$A := \alpha x \text{conjg}(x') + A,$$

where α is a real scalar, x is an n element vector and A is an n by n hermitian matrix.

Parameters
=====

UPLO - CHARACTER*1.
On entry, UPLO specifies whether the upper or lower triangular part of the array A is to be referenced as follows:

UPLO = 'U' or 'u'	Only the upper triangular part of A is to be referenced.
UPLO = 'L' or 'l'	Only the lower triangular part of A is to be referenced.

Unchanged on exit.

N - INTEGER.
On entry, N specifies the order of the matrix A. N must be at least zero.
Unchanged on exit.

ALPHA - REAL.
On entry, ALPHA specifies the scalar α .
Unchanged on exit.

X - COMPLEX array of dimension at least $(1 + (n - 1) * \text{abs}(\text{INCX}))$.
Before entry, the incremented array X must contain the n element vector x .
Unchanged on exit.

INCX - INTEGER.
On entry, INCX specifies the increment for the elements of X. INCX must not be zero.
Unchanged on exit.

A - COMPLEX array of DIMENSION (LDA, n).
 Before entry with UPLO = 'U' or 'u', the leading n by n
 upper triangular part of the array A must contain the upper
 triangular part of the hermitian matrix and the strictly
 lower triangular part of A is not referenced. On exit, the
 upper triangular part of the array A is overwritten by the
 upper triangular part of the updated matrix.
 Before entry with UPLO = 'L' or 'l', the leading n by n
 lower triangular part of the array A must contain the lower
 triangular part of the hermitian matrix and the strictly
 upper triangular part of A is not referenced. On exit, the
 lower triangular part of the array A is overwritten by the
 lower triangular part of the updated matrix.
 Note that the imaginary parts of the diagonal elements need
 not be set, they are assumed to be zero, and on exit they
 are set to zero.

LDA - INTEGER.
 On entry, LDA specifies the first dimension of A as declared
 in the calling (sub) program. LDA must be at least
 max(1, n).
 Unchanged on exit.

***REFERENCES Dongarra, J. J., Du Croz, J., Hammarling, S., and
 Hanson, R. J. An extended set of Fortran basic linear
 algebra subprograms. ACM TOMS, Vol. 14, No. 1,
 pp. 1-17, March 1988.

***ROUTINES CALLED LSAME, XERBLA

***REVISION HISTORY (YYMMDD)
 861022 DATE WRITTEN
 910605 Modified to meet SLATEC prologue standards. Only comment
 lines were modified. (BKS)
 END PROLOGUE

CHER2

```
SUBROUTINE CHER2 (UPLO, N, ALPHA, X, INCX, Y, INCY, A, LDA)
***BEGIN PROLOGUE  CHER2
***PURPOSE  Perform Hermitian rank 2 update of a complex Hermitian
            matrix.
***LIBRARY    SLATEC (BLAS)
***CATEGORY   D1B4
***TYPE       COMPLEX (SHER2-S, DHER2-D, CHER2-C)
***KEYWORDS   LEVEL 2 BLAS, LINEAR ALGEBRA
***AUTHOR    Dongarra, J. J., (ANL)
            Du Croz, J., (NAG)
            Hammarling, S., (NAG)
            Hanson, R. J., (SNLA)
***DESCRIPTION
```

CHER2 performs the hermitian rank 2 operation

$$A := \alpha x \text{conjg}(y') + \text{conjg}(\alpha) y \text{conjg}(x') + A,$$

where alpha is a scalar, x and y are n element vectors and A is an n by n hermitian matrix.

Parameters
=====

UPLO - CHARACTER*1.
On entry, UPLO specifies whether the upper or lower triangular part of the array A is to be referenced as follows:

UPLO = 'U' or 'u'	Only the upper triangular part of A is to be referenced.
UPLO = 'L' or 'l'	Only the lower triangular part of A is to be referenced.

Unchanged on exit.

N - INTEGER.
On entry, N specifies the order of the matrix A.
N must be at least zero.
Unchanged on exit.

ALPHA - COMPLEX .
On entry, ALPHA specifies the scalar alpha.
Unchanged on exit.

X - COMPLEX array of dimension at least
(1 + (n - 1) * abs(INCX)).
Before entry, the incremented array X must contain the n element vector x.
Unchanged on exit.

INCX - INTEGER.
On entry, INCX specifies the increment for the elements of X. INCX must not be zero.
Unchanged on exit.

Y - COMPLEX array of dimension at least
 (1 + (n - 1) * abs(INCY)).
 Before entry, the incremented array Y must contain the n
 element vector y.
 Unchanged on exit.

INCY - INTEGER.
 On entry, INCY specifies the increment for the elements of
 Y. INCY must not be zero.
 Unchanged on exit.

A - COMPLEX array of DIMENSION (LDA, n).
 Before entry with UPLO = 'U' or 'u', the leading n by n
 upper triangular part of the array A must contain the upper
 triangular part of the hermitian matrix and the strictly
 lower triangular part of A is not referenced. On exit, the
 upper triangular part of the array A is overwritten by the
 upper triangular part of the updated matrix.
 Before entry with UPLO = 'L' or 'l', the leading n by n
 lower triangular part of the array A must contain the lower
 triangular part of the hermitian matrix and the strictly
 upper triangular part of A is not referenced. On exit, the
 lower triangular part of the array A is overwritten by the
 lower triangular part of the updated matrix.
 Note that the imaginary parts of the diagonal elements need
 not be set, they are assumed to be zero, and on exit they
 are set to zero.

LDA - INTEGER.
 On entry, LDA specifies the first dimension of A as declared
 in the calling (sub) program. LDA must be at least
 max(1, n).
 Unchanged on exit.

***REFERENCES Dongarra, J. J., Du Croz, J., Hammarling, S., and
 Hanson, R. J. An extended set of Fortran basic linear
 algebra subprograms. ACM TOMS, Vol. 14, No. 1,
 pp. 1-17, March 1988.

***ROUTINES CALLED LSAME, XERBLA

***REVISION HISTORY (YYMMDD)
 861022 DATE WRITTEN
 910605 Modified to meet SLATEC prologue standards. Only comment
 lines were modified. (BKS)
 END PROLOGUE

CHER2K

```
      SUBROUTINE CHER2K (UPLO, TRANS, N, K, ALPHA, A, LDA, B, LDB, BETA,
$      C, LDC)
***BEGIN PROLOGUE  CHER2K
***PURPOSE  Perform Hermitian rank 2k update of a complex.
***LIBRARY    SLATEC (BLAS)
***CATEGORY   D1B6
***TYPE       COMPLEX (SHER2-S, DHER2-D, CHER2-C, CHER2K-C)
***KEYWORDS   LEVEL 3 BLAS, LINEAR ALGEBRA
***AUTHOR    Dongarra, J., (ANL)
              Duff, I., (AERE)
              Du Croz, J., (NAG)
              Hammarling, S. (NAG)
***DESCRIPTION
```

CHER2K performs one of the hermitian rank 2k operations

$$C := \alpha A \text{conjg}(B') + \text{conjg}(\alpha) B \text{conjg}(A') + \beta C,$$

or

$$C := \alpha \text{conjg}(A') B + \text{conjg}(\alpha) \text{conjg}(B') A + \beta C,$$

where α and β are scalars with β real, C is an n by n hermitian matrix and A and B are n by k matrices in the first case and k by n matrices in the second case.

Parameters
=====

UPLO - CHARACTER*1.

On entry, UPLO specifies whether the upper or lower triangular part of the array C is to be referenced as follows:

UPLO = 'U' or 'u' Only the upper triangular part of C is to be referenced.

UPLO = 'L' or 'l' Only the lower triangular part of C is to be referenced.

Unchanged on exit.

TRANS - CHARACTER*1.

On entry, TRANS specifies the operation to be performed as follows:

TRANS = 'N' or 'n' $C := \alpha A \text{conjg}(B') + \text{conjg}(\alpha) B \text{conjg}(A') + \beta C.$

TRANS = 'C' or 'c' $C := \alpha \text{conjg}(A') B + \text{conjg}(\alpha) \text{conjg}(B') A + \beta C.$

Unchanged on exit.

- N - INTEGER.
On entry, N specifies the order of the matrix C. N must be at least zero.
Unchanged on exit.
- K - INTEGER.
On entry with TRANS = 'N' or 'n', K specifies the number of columns of the matrices A and B, and on entry with TRANS = 'C' or 'c', K specifies the number of rows of the matrices A and B. K must be at least zero.
Unchanged on exit.
- ALPHA - COMPLEX .
On entry, ALPHA specifies the scalar alpha.
Unchanged on exit.
- A - COMPLEX array of DIMENSION (LDA, ka), where ka is k when TRANS = 'N' or 'n', and is n otherwise.
Before entry with TRANS = 'N' or 'n', the leading n by k part of the array A must contain the matrix A, otherwise the leading k by n part of the array A must contain the matrix A.
Unchanged on exit.
- LDA - INTEGER.
On entry, LDA specifies the first dimension of A as declared in the calling (sub) program. When TRANS = 'N' or 'n' then LDA must be at least max(1, n), otherwise LDA must be at least max(1, k).
Unchanged on exit.
- B - COMPLEX array of DIMENSION (LDB, kb), where kb is k when TRANS = 'N' or 'n', and is n otherwise.
Before entry with TRANS = 'N' or 'n', the leading n by k part of the array B must contain the matrix B, otherwise the leading k by n part of the array B must contain the matrix B.
Unchanged on exit.
- LDB - INTEGER.
On entry, LDB specifies the first dimension of B as declared in the calling (sub) program. When TRANS = 'N' or 'n' then LDB must be at least max(1, n), otherwise LDB must be at least max(1, k).
Unchanged on exit.
- BETA - REAL .
On entry, BETA specifies the scalar beta.
Unchanged on exit.
- C - COMPLEX array of DIMENSION (LDC, n).
Before entry with UPLO = 'U' or 'u', the leading n by n upper triangular part of the array C must contain the upper triangular part of the hermitian matrix and the strictly lower triangular part of C is not referenced. On exit, the upper triangular part of the array C is overwritten by the upper triangular part of the updated matrix.
Before entry with UPLO = 'L' or 'l', the leading n by n lower triangular part of the array C must contain the lower triangular part of the hermitian matrix and the strictly

upper triangular part of C is not referenced. On exit, the lower triangular part of the array C is overwritten by the lower triangular part of the updated matrix.
Note that the imaginary parts of the diagonal elements need not be set, they are assumed to be zero, and on exit they are set to zero.

LDC - INTEGER.

On entry, LDC specifies the first dimension of C as declared in the calling (sub) program. LDC must be at least $\max(1, n)$.
Unchanged on exit.

***REFERENCES Dongarra, J., Du Croz, J., Duff, I., and Hammarling, S.
A set of level 3 basic linear algebra subprograms.
ACM TOMS, Vol. 16, No. 1, pp. 1-17, March 1990.

***ROUTINES CALLED LSAME, XERBLA

***REVISION HISTORY (YYMMDD)

890208 DATE WRITTEN

910605 Modified to meet SLATEC prologue standards. Only comment lines were modified. (BKS)

END PROLOGUE

CHERK

```
      SUBROUTINE CHERK (UPLO, TRANS, N, K, ALPHA, A, LDA, BETA, C, LDC)
***BEGIN PROLOGUE  CHERK
***PURPOSE  Perform Hermitian rank k update of a complex Hermitian
            matrix.
***LIBRARY   SLATEC (BLAS)
***CATEGORY  D1B6
***TYPE      COMPLEX (SHERK-S, DHERK-D, CHERK-C)
***KEYWORDS  LEVEL 3 BLAS, LINEAR ALGEBRA
***AUTHOR    Dongarra, J., (ANL)
            Duff, I., (AERE)
            Du Croz, J., (NAG)
            Hammarling, S. (NAG)
***DESCRIPTION
```

CHERK performs one of the hermitian rank k operations

$$C := \alpha A \text{conjg}(A') + \beta C,$$

or

$$C := \alpha \text{conjg}(A') A + \beta C,$$

where α and β are real scalars, C is an n by n hermitian matrix and A is an n by k matrix in the first case and a k by n matrix in the second case.

Parameters
=====

UPLO - CHARACTER*1.
On entry, UPLO specifies whether the upper or lower triangular part of the array C is to be referenced as follows:

UPLO = 'U' or 'u' Only the upper triangular part of C is to be referenced.

UPLO = 'L' or 'l' Only the lower triangular part of C is to be referenced.

Unchanged on exit.

TRANS - CHARACTER*1.
On entry, TRANS specifies the operation to be performed as follows:

TRANS = 'N' or 'n' $C := \alpha A \text{conjg}(A') + \beta C.$

TRANS = 'C' or 'c' $C := \alpha \text{conjg}(A') A + \beta C.$

Unchanged on exit.

N - INTEGER.
On entry, N specifies the order of the matrix C . N must be at least zero.
Unchanged on exit.

K - INTEGER.
On entry with TRANS = 'N' or 'n', K specifies the number of columns of the matrix A, and on entry with TRANS = 'C' or 'c', K specifies the number of rows of the matrix A. K must be at least zero.
Unchanged on exit.

ALPHA - REAL .
On entry, ALPHA specifies the scalar alpha.
Unchanged on exit.

A - COMPLEX array of DIMENSION (LDA, ka), where ka is k when TRANS = 'N' or 'n', and is n otherwise.
Before entry with TRANS = 'N' or 'n', the leading n by k part of the array A must contain the matrix A, otherwise the leading k by n part of the array A must contain the matrix A.
Unchanged on exit.

LDA - INTEGER.
On entry, LDA specifies the first dimension of A as declared in the calling (sub) program. When TRANS = 'N' or 'n' then LDA must be at least max(1, n), otherwise LDA must be at least max(1, k).
Unchanged on exit.

BETA - REAL .
On entry, BETA specifies the scalar beta.
Unchanged on exit.

C - COMPLEX array of DIMENSION (LDC, n).
Before entry with UPLO = 'U' or 'u', the leading n by n upper triangular part of the array C must contain the upper triangular part of the hermitian matrix and the strictly lower triangular part of C is not referenced. On exit, the upper triangular part of the array C is overwritten by the upper triangular part of the updated matrix.
Before entry with UPLO = 'L' or 'l', the leading n by n lower triangular part of the array C must contain the lower triangular part of the hermitian matrix and the strictly upper triangular part of C is not referenced. On exit, the lower triangular part of the array C is overwritten by the lower triangular part of the updated matrix.
Note that the imaginary parts of the diagonal elements need not be set, they are assumed to be zero, and on exit they are set to zero.

LDC - INTEGER.
On entry, LDC specifies the first dimension of C as declared in the calling (sub) program. LDC must be at least max(1, n).
Unchanged on exit.

***REFERENCES Dongarra, J., Du Croz, J., Duff, I., and Hammarling, S.
A set of level 3 basic linear algebra subprograms.
ACM TOMS, Vol. 16, No. 1, pp. 1-17, March 1990.

***ROUTINES CALLED LSAME, XERBLA

***REVISION HISTORY (YYMMDD)
890208 DATE WRITTEN

910605 Modified to meet SLATEC prologue standards. Only comment
lines were modified. (BKS)
END PROLOGUE

CHFDV

```
      SUBROUTINE CHFDV (X1, X2, F1, F2, D1, D2, NE, XE, FE, DE, NEXT,
+      IERR)
***BEGIN PROLOGUE  CHFDV
***PURPOSE  Evaluate a cubic polynomial given in Hermite form and its
             first derivative at an array of points.  While designed for
             use by PCHFD, it may be useful directly as an evaluator
             for a piecewise cubic Hermite function in applications,
             such as graphing, where the interval is known in advance.
             If only function values are required, use CHFEV instead.
***LIBRARY    SLATEC (PCHIP)
***CATEGORY   E3, H1
***TYPE       SINGLE PRECISION (CHFDV-S, DCHFDV-D)
***KEYWORDS   CUBIC HERMITE DIFFERENTIATION, CUBIC HERMITE EVALUATION,
             CUBIC POLYNOMIAL EVALUATION, PCHIP
***AUTHOR     Fritsch, F. N., (LLNL)
             Lawrence Livermore National Laboratory
             P.O. Box 808 (L-316)
             Livermore, CA 94550
             FTS 532-4275, (510) 422-4275
***DESCRIPTION
```

CHFDV: Cubic Hermite Function and Derivative Evaluator

Evaluates the cubic polynomial determined by function values F1,F2 and derivatives D1,D2 on interval (X1,X2), together with its first derivative, at the points XE(J), J=1(1)NE.

If only function values are required, use CHFEV, instead.

Calling sequence:

```
      INTEGER  NE, NEXT(2), IERR
      REAL     X1, X2, F1, F2, D1, D2, XE(NE), FE(NE), DE(NE)

      CALL  CHFDV (X1,X2, F1,F2, D1,D2, NE, XE, FE, DE, NEXT, IERR)
```

Parameters:

X1,X2 -- (input) endpoints of interval of definition of cubic.
(Error return if X1.EQ.X2 .)

F1,F2 -- (input) values of function at X1 and X2, respectively.

D1,D2 -- (input) values of derivative at X1 and X2, respectively.

NE -- (input) number of evaluation points. (Error return if
NE.LT.1 .)

XE -- (input) real array of points at which the functions are to
be evaluated. If any of the XE are outside the interval
[X1,X2], a warning error is returned in NEXT.

FE -- (output) real array of values of the cubic function defined
by X1,X2, F1,F2, D1,D2 at the points XE.

```

DE -- (output) real array of values of the first derivative of
      the same function at the points  XE.

NEXT -- (output) integer array indicating number of extrapolation
      points:
      NEXT(1) = number of evaluation points to left of interval.
      NEXT(2) = number of evaluation points to right of interval.

IERR -- (output) error flag.
      Normal return:
      IERR = 0  (no errors).
      "Recoverable" errors:
      IERR = -1  if NE.LT.1 .
      IERR = -2  if X1.EQ.X2 .
      (Output arrays have not been changed in either case.)

***REFERENCES  (NONE)
***ROUTINES CALLED  XERMSG
***REVISION HISTORY  (YYMMDD)
811019  DATE WRITTEN
820803  Minor cosmetic changes for release 1.
890411  Added SAVE statements (Vers. 3.2).
890531  Changed all specific intrinsics to generic.  (WRB)
890831  Modified array declarations.  (WRB)
890831  REVISION DATE from Version 3.2
891214  Prologue converted to Version 4.0 format.  (BAB)
900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
END PROLOGUE

```

CHFEV

```
SUBROUTINE CHFEV (X1, X2, F1, F2, D1, D2, NE, XE, FE, NEXT, IERR)
***BEGIN PROLOGUE  CHFEV
***PURPOSE  Evaluate a cubic polynomial given in Hermite form at an
             array of points.  While designed for use by PCHFE, it may
             be useful directly as an evaluator for a piecewise cubic
             Hermite function in applications, such as graphing, where
             the interval is known in advance.
***LIBRARY   SLATEC (PCHIP)
***CATEGORY  E3
***TYPE      SINGLE PRECISION (CHFEV-S, DCHFEV-D)
***KEYWORDS  CUBIC HERMITE EVALUATION, CUBIC POLYNOMIAL EVALUATION,
             PCHIP
***AUTHOR    Fritsch, F. N., (LLNL)
             Lawrence Livermore National Laboratory
             P.O. Box 808  (L-316)
             Livermore, CA  94550
             FTS 532-4275, (510) 422-4275
***DESCRIPTION
```

CHFEV: Cubic Hermite Function Evaluator

Evaluates the cubic polynomial determined by function values
F1,F2 and derivatives D1,D2 on interval (X1,X2) at the points
XE(J), J=1(1)NE.

Calling sequence:

```
INTEGER  NE, NEXT(2), IERR
REAL     X1, X2, F1, F2, D1, D2, XE(NE), FE(NE)

CALL  CHFEV (X1,X2, F1,F2, D1,D2, NE, XE, FE, NEXT, IERR)
```

Parameters:

X1,X2 -- (input) endpoints of interval of definition of cubic.
(Error return if X1.EQ.X2 .)

F1,F2 -- (input) values of function at X1 and X2, respectively.

D1,D2 -- (input) values of derivative at X1 and X2, respectively.

NE -- (input) number of evaluation points. (Error return if
NE.LT.1 .)

XE -- (input) real array of points at which the function is to be
evaluated. If any of the XE are outside the interval
[X1,X2], a warning error is returned in NEXT.

FE -- (output) real array of values of the cubic function defined
by X1,X2, F1,F2, D1,D2 at the points XE.

NEXT -- (output) integer array indicating number of extrapolation
points:
NEXT(1) = number of evaluation points to left of interval.

```

        NEXT(2) = number of evaluation points to right of interval.

IERR -- (output) error flag.
      Normal return:
        IERR = 0  (no errors).
      "Recoverable" errors:
        IERR = -1  if NE.LT.1 .
        IERR = -2  if X1.EQ.X2 .
        (The FE-array has not been changed in either case.)

***REFERENCES  (NONE)
***ROUTINES CALLED  XERMSG
***REVISION HISTORY  (YYMMDD)
      811019  DATE WRITTEN
      820803  Minor cosmetic changes for release 1.
      890411  Added SAVE statements (Vers. 3.2).
      890531  Changed all specific intrinsics to generic.  (WRB)
      890703  Corrected category record.  (WRB)
      890703  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
END PROLOGUE

```

CHICO

```
SUBROUTINE CHICO (A, LDA, N, KPVT, RCOND, Z)
***BEGIN PROLOGUE  CHICO
***PURPOSE  Factor a complex Hermitian matrix by elimination with sym-
             metric pivoting and estimate the condition of the matrix.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2D1A
***TYPE      COMPLEX (SSICO-S, DSICO-D, CHICO-C, CSICO-C)
***KEYWORDS  CONDITION NUMBER, HERMITIAN, LINEAR ALGEBRA, LINPACK,
             MATRIX FACTORIZATION
***AUTHOR  Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CHICO factors a complex Hermitian matrix by elimination with symmetric pivoting and estimates the condition of the matrix.

If RCOND is not needed, CHIFA is slightly faster.
To solve $A \cdot X = B$, follow CHICO by CHISL.
To compute $\text{INVERSE}(A) \cdot C$, follow CHICO by CHISL.
To compute $\text{INVERSE}(A)$, follow CHICO by CHIDI.
To compute $\text{DETERMINANT}(A)$, follow CHICO by CHIDI.
To compute $\text{INERTIA}(A)$, follow CHICO by CHIDI.

On Entry

A COMPLEX(LDA, N)
 the Hermitian matrix to be factored.
 Only the diagonal and upper triangle are used.

LDA INTEGER
 the leading dimension of the array A .

N INTEGER
 the order of the matrix A .

Output

A a block diagonal matrix and the multipliers which
 were used to obtain it.
 The factorization can be written $A = U \cdot D \cdot \text{CTRANS}(U)$
 where U is a product of permutation and unit
 upper triangular matrices, CTRANS(U) is the
 conjugate transpose of U, and D is block diagonal
 with 1 by 1 and 2 by 2 blocks.

KVPT INTEGER(N)
 an integer vector of pivot indices.

RCOND REAL
 an estimate of the reciprocal condition of A .
 For the system $A \cdot X = B$, relative perturbations
 in A and B of size EPSILON may cause
 relative perturbations in X of size $\text{EPSILON}/\text{RCOND}$.
 If RCOND is so small that the logical expression
 $1.0 + \text{RCOND} \cdot \text{EQ} \cdot 1.0$
 is true, then A may be singular to working
 precision. In particular, RCOND is zero if

exact singularity is detected or the estimate underflows.

Z COMPLEX(N)
 a work vector whose contents are usually unimportant.
 If A is close to a singular matrix, then Z is
 an approximate null vector in the sense that
 $\text{NORM}(A*Z) = \text{RCOND}*\text{NORM}(A)*\text{NORM}(Z)$.

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
 Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CAXPY, CDOTC, CHIFA, CSSCAL, SCASUM

***REVISION HISTORY (YYMMDD)

780814 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

891107 Modified routine equivalence list. (WRB)

891107 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900326 Removed duplicate information from DESCRIPTION section.
(WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CHIDI

```
SUBROUTINE CHIDI (A, LDA, N, KPVT, DET, INERT, WORK, JOB)
***BEGIN PROLOGUE  CHIDI
***PURPOSE  Compute the determinant, inertia and inverse of a complex
             Hermitian matrix using the factors obtained from CHIFA.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2D1A, D3D1A
***TYPE      COMPLEX (SSIDI-S, DSISI-D, CHIDI-C, CSIDI-C)
***KEYWORDS  DETERMINANT, HERMITIAN, INVERSE, LINEAR ALGEBRA, LINPACK,
             MATRIX
***AUTHOR    Bunch, J., (UCSD)
***DESCRIPTION
```

CHIDI computes the determinant, inertia and inverse of a complex Hermitian matrix using the factors from CHIFA.

On Entry

A COMPLEX(LDA,N)
 the output from CHIFA.

LDA INTEGER
 the leading dimension of the array A.

N INTEGER
 the order of the matrix A.

KPVT INTEGER(N)
 the pivot vector from CHIFA.

WORK COMPLEX(N)
 work vector. Contents destroyed.

JOB INTEGER
 JOB has the decimal expansion ABC where
 if C .NE. 0, the inverse is computed,
 if B .NE. 0, the determinant is computed,
 if A .NE. 0, the inertia is computed.

 For example, JOB = 111 gives all three.

On Return

Variables not requested by JOB are not used.

A contains the upper triangle of the inverse of
 the original matrix. The strict lower triangle
 is never referenced.

DET REAL(2)
 determinant of original matrix.
 Determinant = DET(1) * 10.0**DET(2)
 with 1.0 .LE. ABS(DET(1)) .LT. 10.0
 or DET(1) = 0.0.

INERT INTEGER(3)
 the inertia of the original matrix.

INERT(1) = number of positive eigenvalues.
INERT(2) = number of negative eigenvalues.
INERT(3) = number of zero eigenvalues.

Error Condition

A division by zero may occur if the inverse is requested
and CHICO has set RCOND .EQ. 0.0
or CHIFA has set INFO .NE. 0 .

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CAXPY, CCOPY, CDOTC, CSWAP

***REVISION HISTORY (YYMMDD)

780814 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

891107 Modified routine equivalence list. (WRB)

891107 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900326 Removed duplicate information from DESCRIPTION section.
(WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CHIEV

```
SUBROUTINE CHIEV (A, LDA, N, E, V, LDV, WORK, JOB, INFO)
***BEGIN PROLOGUE  CHIEV
***PURPOSE  Compute the eigenvalues and, optionally, the eigenvectors
             of a complex Hermitian matrix.
***LIBRARY   SLATEC
***CATEGORY  D4A3
***TYPE      COMPLEX (SSIEV-S, CHIEV-C)
***KEYWORDS  COMPLEX HERMITIAN, EIGENVALUES, EIGENVECTORS, MATRIX,
             SYMMETRIC
***AUTHOR    Kahaner, D. K., (NBS)
             Moler, C. B., (U. of New Mexico)
             Stewart, G. W., (U. of Maryland)
***DESCRIPTION
```

David Kahaner, Cleve Moler, G. W. Stewart,
N.B.S. U.N.M. N.B.S./U.MD.

Abstract

CHIEV computes the eigenvalues and, optionally,
the eigenvectors of a complex Hermitian matrix.

Call Sequence Parameters-

(the values of parameters marked with * (star) will be changed
by CHIEV.)

A*	COMPLEX(LDA,N) complex Hermitian input matrix. Only the upper triangle of A need be filled in. Elements on diagonal must be real.
LDA	INTEGER set by the user to the leading dimension of the complex array A.
N	INTEGER set by the user to the order of the matrices A and V, and the number of elements in E.
E*	REAL(N) on return from CHIEV E contains the eigenvalues of A. See also INFO below.
V*	COMPLEX(LDV,N) on return from CHIEV if the user has set JOB = 0 V is not referenced. = nonzero the N eigenvectors of A are stored in the first N columns of V. See also INFO below.
LDV	INTEGER set by the user to the leading dimension of the array V if JOB is also set nonzero. In that case N must be .LE. LDV. If JOB is set to zero LDV is not referenced.
WORK*	REAL(4N)

temporary storage vector. Contents changed by CHIEV.

JOB INTEGER
 set by the user to
 = 0 eigenvalues only to be calculated by CHIEV.
 Neither V nor LDV are referenced.
 = nonzero eigenvalues and vectors to be calculated.
 In this case A and V must be distinct arrays
 also if LDA .GT. LDV CHIEV changes all the
 elements of A thru column N. If LDA < LDV
 CHIEV changes all the elements of V through
 column N. If LDA = LDV only A(I,J) and V(I,
 J) for I,J = 1,...,N are changed by CHIEV.

INFO* INTEGER
 on return from CHIEV the value of INFO is
 = 0 normal return, calculation successful.
 = K if the eigenvalue iteration fails to converge,
 eigenvalues (and eigenvectors if requested)
 1 through K-1 are correct.

Error Messages

No. 1 recoverable N is greater than LDA
No. 2 recoverable N is less than one.
No. 3 recoverable JOB is nonzero and N is greater than LDV
No. 4 warning LDA > LDV, elements of A other than the
 N by N input elements have been changed
No. 5 warning LDA < LDV, elements of V other than the
 N by N output elements have been changed
No. 6 recoverable nonreal element on diagonal of A.

***REFERENCES (NONE)

***ROUTINES CALLED HTRIBK, HTRIDI, IMTQL2, SCOPY, SCOPYM, TQLRAT,
 XERMSG

***REVISION HISTORY (YYMMDD)

800808 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890531 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900315 CALLS to XERROR changed to CALLS to XERMSG. (THJ)

END PROLOGUE

CHIFA

```
SUBROUTINE CHIFA (A, LDA, N, KPVT, INFO)
***BEGIN PROLOGUE  CHIFA
***PURPOSE  Factor a complex Hermitian matrix by elimination
             (symmetric pivoting).
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2D1A
***TYPE      COMPLEX (SSIFA-S, DSIFA-D, CHIFA-C, CSIFA-C)
***KEYWORDS  HERMITIAN, LINEAR ALGEBRA, LINPACK, MATRIX FACTORIZATION
***AUTHOR   Bunch, J., (UCSD)
***DESCRIPTION
```

CHIFA factors a complex Hermitian matrix by elimination with symmetric pivoting.

To solve $A \cdot X = B$, follow CHIFA by CHISL.
To compute $\text{INVERSE}(A) \cdot C$, follow CHIFA by CHISL.
To compute $\text{DETERMINANT}(A)$, follow CHIFA by CHIDI.
To compute $\text{INERTIA}(A)$, follow CHIFA by CHIDI.
To compute $\text{INVERSE}(A)$, follow CHIFA by CHIDI.

On Entry

A COMPLEX(LDA,N)
 the Hermitian matrix to be factored.
 Only the diagonal and upper triangle are used.

LDA INTEGER
 the leading dimension of the array A .

N INTEGER
 the order of the matrix A .

On Return

A a block diagonal matrix and the multipliers which were used to obtain it.
 The factorization can be written $A = U \cdot D \cdot \text{CTRANS}(U)$ where U is a product of permutation and unit upper triangular matrices , $\text{CTRANS}(U)$ is the conjugate transpose of U , and D is block diagonal with 1 by 1 and 2 by 2 blocks.

KVPT INTEGER(N)
 an integer vector of pivot indices.

INFO INTEGER
 = 0 normal value.
 = K if the K-th pivot block is singular. This is not an error condition for this subroutine, but it does indicate that CHISL or CHIDI may divide by zero if called.

```
***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
                Stewart, LINPACK Users' Guide, SIAM, 1979.
***ROUTINES CALLED  CAXPY, CSWAP, ICAMAX
***REVISION HISTORY  (YYMMDD)
```

780814 DATE WRITTEN
890531 Changed all specific intrinsics to generic. (WRB)
890831 Modified array declarations. (WRB)
891107 Modified routine equivalence list. (WRB)
891107 REVISION DATE from Version 3.2
891214 Prologue converted to Version 4.0 format. (BAB)
900326 Removed duplicate information from DESCRIPTION section.
(WRB)
920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

CHISL

```
      SUBROUTINE CHISL (A, LDA, N, KPVT, B)
***BEGIN PROLOGUE  CHISL
***PURPOSE  Solve the complex Hermitian system using factors obtained
            from CHIFA.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2D1A
***TYPE      COMPLEX (SSISL-S, DSISL-D, CHISL-C, CSISL-C)
***KEYWORDS  HERMITIAN, LINEAR ALGEBRA, LINPACK, MATRIX, SOLVE
***AUTHOR   Bunch, J., (UCSD)
***DESCRIPTION
```

CHISL solves the complex Hermitian system
 $A * X = B$
using the factors computed by CHIFA.

On Entry

A COMPLEX(LDA,N)
 the output from CHIFA.

LDA INTEGER
 the leading dimension of the array A .

N INTEGER
 the order of the matrix A .

KVPT INTEGER(N)
 the pivot vector from CHIFA.

B COMPLEX(N)
 the right hand side vector.

On Return

B the solution vector X .

Error Condition

A division by zero may occur if CHICO has set RCOND .EQ. 0.0
or CHIFA has set INFO .NE. 0 .

To compute $INVERSE(A) * C$ where C is a matrix
with P columns

```
      CALL CHIFA(A,LDA,N,KVPT,INFO)
      IF (INFO .NE. 0) GO TO ...
      DO 10 J = 1, p
        CALL CHISL(A,LDA,N,KVPT,C(1,J))
      10 CONTINUE
```

```
***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
               Stewart, LINPACK Users' Guide, SIAM, 1979.
***ROUTINES CALLED  CAXPY, CDOTC
***REVISION HISTORY  (YYMMDD)
      780814  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890831  Modified array declarations.  (WRB)
```

891107 Modified routine equivalence list. (WRB)
891107 REVISION DATE from Version 3.2
891214 Prologue converted to Version 4.0 format. (BAB)
900326 Removed duplicate information from DESCRIPTION section.
(WRB)
920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

CHKDER

```
      SUBROUTINE CHKDER (M, N, X, FVEC, FJAC, LDFJAC, XP, FVECP, MODE,  
+      ERR)  
***BEGIN PROLOGUE  CHKDER  
***PURPOSE  Check the gradients of M nonlinear functions in N  
             variables, evaluated at a point X, for consistency  
             with the functions themselves.  
***LIBRARY    SLATEC  
***CATEGORY   F3, G4C  
***TYPE       SINGLE PRECISION (CHKDER-S, DCKDER-D)  
***KEYWORDS   GRADIENTS, JACOBIAN, MINPACK, NONLINEAR  
***AUTHOR    Hiebert, K. L. (SNLA)  
***DESCRIPTION
```

This subroutine is a companion routine to SNLS1, SNLS1E, SNSQ, and SNSQE which may be used to check the calculation of the Jacobian.

SUBROUTINE CHKDER

This subroutine checks the gradients of M nonlinear functions in N variables, evaluated at a point X, for consistency with the functions themselves. The user must call CKDER twice, first with MODE = 1 and then with MODE = 2.

MODE = 1. On input, X must contain the point of evaluation.
On output, XP is set to a neighboring point.

MODE = 2. On input, FVEC must contain the functions and the rows of FJAC must contain the gradients of the respective functions each evaluated at X, and FVECP must contain the functions evaluated at XP.
On output, ERR contains measures of correctness of the respective gradients.

The subroutine does not perform reliably if cancellation or rounding errors cause a severe loss of significance in the evaluation of a function. Therefore, none of the components of X should be unusually small (in particular, zero) or any other value which may cause loss of significance.

The SUBROUTINE statement is

```
      SUBROUTINE CHKDER(M,N,X,FVEC,FJAC,LDFJAC,XP,FVECP,MODE,ERR)
```

where

M is a positive integer input variable set to the number of functions.

N is a positive integer input variable set to the number of variables.

X is an input array of length N.

FVEC is an array of length M. On input when MODE = 2, FVEC must contain the functions evaluated at X.

FJAC is an M by N array. On input when MODE = 2, the rows of FJAC must contain the gradients of the respective functions evaluated at X.

LDFJAC is a positive integer input parameter not less than M which specifies the leading dimension of the array FJAC.

XP is an array of length N. On output when MODE = 1, XP is set to a neighboring point of X.

FVECP is an array of length M. On input when MODE = 2, FVECP must contain the functions evaluated at XP.

MODE is an integer input variable set to 1 on the first call and 2 on the second. Other values of MODE are equivalent to MODE = 1.

ERR is an array of length M. On output when MODE = 2, ERR contains measures of correctness of the respective gradients. If there is no severe loss of significance, then if ERR(I) is 1.0 the I-th gradient is correct, while if ERR(I) is 0.0 the I-th gradient is incorrect. For values of ERR between 0.0 and 1.0, the categorization is less certain. In general, a value of ERR(I) greater than 0.5 indicates that the I-th gradient is probably correct, while a value of ERR(I) less than 0.5 indicates that the I-th gradient is probably incorrect.

***REFERENCES M. J. D. Powell, A hybrid method for nonlinear equations. In Numerical Methods for Nonlinear Algebraic Equations, P. Rabinowitz, Editor. Gordon and Breach, 1988.

***ROUTINES CALLED R1MACH

***REVISION HISTORY (YYMMDD)

800301 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900326 Removed duplicate information from DESCRIPTION section. (WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CHPCO

```
SUBROUTINE CHPCO (AP, N, KPVT, RCOND, Z)
***BEGIN PROLOGUE  CHPCO
***PURPOSE  Factor a complex Hermitian matrix stored in packed form by
             elimination with symmetric pivoting and estimate the
             condition number of the matrix.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2D1A
***TYPE      COMPLEX (SSPCO-S, DSPCO-D, CHPCO-C, CSPCO-C)
***KEYWORDS  CONDITION NUMBER, HERMITIAN, LINEAR ALGEBRA, LINPACK,
             MATRIX FACTORIZATION, PACKED
***AUTHOR  Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CHPCO factors a complex Hermitian matrix stored in packed form by elimination with symmetric pivoting and estimates the condition of the matrix.

if RCOND is not needed, CHPFA is slightly faster.
To solve $A \cdot X = B$, follow CHPCO by CHPSL.
To compute $\text{INVERSE}(A) \cdot C$, follow CHPCO by CHPSL.
To compute $\text{INVERSE}(A)$, follow CHPCO by CHPDI.
To compute $\text{DETERMINANT}(A)$, follow CHPCO by CHPDI.
To compute $\text{INERTIA}(A)$, follow CHPCO by CHPDI.

On Entry

AP COMPLEX (N*(N+1)/2)
 the packed form of a Hermitian matrix A. The
 columns of the upper triangle are stored sequentially
 in a one-dimensional array of length N*(N+1)/2.
 See comments below for details.

N INTEGER
 the order of the matrix A.

Output

AP a block diagonal matrix and the multipliers which
 were used to obtain it stored in packed form.
 The factorization can be written $A = U \cdot D \cdot \text{CTRANS}(U)$
 where U is a product of permutation and unit
 upper triangular matrices, CTRANS(U) is the
 conjugate transpose of U, and D is block diagonal
 with 1 by 1 and 2 by 2 blocks.

KVPT INTEGER(N)
 an integer vector of pivot indices.

RCOND REAL
 an estimate of the reciprocal condition of A.
 For the system $A \cdot X = B$, relative perturbations
 in A and B of size EPSILON may cause
 relative perturbations in X of size EPSILON/RCOND.
 If RCOND is so small that the logical expression
 $1.0 + \text{RCOND} \text{ .EQ. } 1.0$
 is true, then A may be singular to working

precision. In particular, RCOND is zero if exact singularity is detected or the estimate underflows.

Z COMPLEX(N)
a work vector whose contents are usually unimportant.
If A is close to a singular matrix, then Z is
an approximate null vector in the sense that
 $\text{NORM}(A*Z) = \text{RCOND}*\text{NORM}(A)*\text{NORM}(Z)$.

Packed Storage

The following program segment will pack the upper triangle of a Hermitian matrix.

```
      K = 0
      DO 20 J = 1, N
        DO 10 I = 1, J
          K = K + 1
          AP(K) = A(I,J)
10      CONTINUE
20      CONTINUE
```

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CAXPY, CDOTC, CHPFA, CSSCAL, SCASUM

***REVISION HISTORY (YYMMDD)

780814 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

891107 Modified routine equivalence list. (WRB)

891107 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900326 Removed duplicate information from DESCRIPTION section.
(WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CHPDI

```
SUBROUTINE CHPDI (AP, N, KPVT, DET, INERT, WORK, JOB)
***BEGIN PROLOGUE  CHPDI
***PURPOSE  Compute the determinant, inertia and inverse of a complex
             Hermitian matrix stored in packed form using the factors
             obtained from CHPFA.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2D1A, D3D1A
***TYPE      COMPLEX (SSPDI-S, DSPDI-D, CHPDI-C, DSPDI-C)
***KEYWORDS  DETERMINANT, HERMITIAN, INVERSE, LINEAR ALGEBRA, LINPACK,
             MATRIX, PACKED
***AUTHOR    Bunch, J., (UCSD)
***DESCRIPTION
```

CHPDI computes the determinant, inertia and inverse of a complex Hermitian matrix using the factors from CHPFA, where the matrix is stored in packed form.

On Entry

AP COMPLEX (N*(N+1)/2)
 the output from CHPFA.

N INTEGER
 the order of the matrix A.

KPVT INTEGER(N)
 the pivot vector from CHPFA.

WORK COMPLEX(N)
 work vector. Contents ignored.

JOB INTEGER
 JOB has the decimal expansion ABC where
 if C .NE. 0, the inverse is computed,
 if B .NE. 0, the determinant is computed,
 if A .NE. 0, the inertia is computed.

 For example, JOB = 111 gives all three.

On Return

Variables not requested by JOB are not used.

AP contains the upper triangle of the inverse of
 the original matrix, stored in packed form.
 The columns of the upper triangle are stored
 sequentially in a one-dimensional array.

DET REAL(2)
 determinant of original matrix.
 Determinant = DET(1) * 10.0**DET(2)
 with 1.0 .LE. ABS(DET(1)) .LT. 10.0
 or DET(1) = 0.0.

INERT INTEGER(3)
 the inertia of the original matrix.

INERT(1) = number of positive eigenvalues.
INERT(2) = number of negative eigenvalues.
INERT(3) = number of zero eigenvalues.

Error Condition

A division by zero will occur if the inverse is requested
and CHPCO has set RCOND .EQ. 0.0
or CHPFA has set INFO .NE. 0 .

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CAXPY, CCOPY, CDOTC, CSWAP

***REVISION HISTORY (YYMMDD)

780814 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

891107 Modified routine equivalence list. (WRB)

891107 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900326 Removed duplicate information from DESCRIPTION section.
(WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CHPFA

```
SUBROUTINE CHPFA (AP, N, KPVT, INFO)
***BEGIN PROLOGUE  CHPFA
***PURPOSE  Factor a complex Hermitian matrix stored in packed form by
             elimination with symmetric pivoting.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2D1A
***TYPE      COMPLEX (SSPFA-S, DSPFA-D, CHPFA-C, DSPFA-C)
***KEYWORDS  HERMITIAN, LINEAR ALGEBRA, LINPACK, MATRIX FACTORIZATION,
             PACKED
***AUTHOR   Bunch, J., (UCSD)
***DESCRIPTION
```

CHPFA factors a complex Hermitian matrix stored in packed form by elimination with symmetric pivoting.

To solve $A^*X = B$, follow CHPFA by CHPSL.
To compute $INVERSE(A)^*C$, follow CHPFA by CHPSL.
To compute $DETERMINANT(A)$, follow CHPFA by CHPDI.
To compute $INERTIA(A)$, follow CHPFA by CHPDI.
To compute $INVERSE(A)$, follow CHPFA by CHPDI.

On Entry

AP COMPLEX (N*(N+1)/2)
 the packed form of a Hermitian matrix A . The
 columns of the upper triangle are stored sequentially
 in a one-dimensional array of length N*(N+1)/2 .
 See comments below for details.

N INTEGER
 the order of the matrix A .

Output

AP A block diagonal matrix and the multipliers which
 were used to obtain it stored in packed form.
 The factorization can be written $A = U^*D^*CTRANS(U)$
 where U is a product of permutation and unit
 upper triangular matrices , CTRANS(U) is the
 conjugate transpose of U , and D is block diagonal
 with 1 by 1 and 2 by 2 blocks.

KPVT INTEGER(N)
 an integer vector of pivot indices.

INFO INTEGER
 = 0 normal value.
 = K if the K-th pivot block is singular. This is
 not an error condition for this subroutine,
 but it does indicate that CHPSL or CHPDI may
 divide by zero if called.

Packed Storage

The following program segment will pack the upper
triangle of a Hermitian matrix.

```

      K = 0
      DO 20 J = 1, N
        DO 10 I = 1, J
          K = K + 1
          AP(K) = A(I,J)
10      CONTINUE
20 CONTINUE

```

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CAXPY, CSWAP, ICAMAX

***REVISION HISTORY (YYMMDD)

780814 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

891107 Modified routine equivalence list. (WRB)

891107 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900326 Removed duplicate information from DESCRIPTION section.
(WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CHPMV

```
      SUBROUTINE CHPMV (UPLO, N, ALPHA, AP, X, INCX, BETA, Y, INCY)
***BEGIN PROLOGUE  CHPMV
***PURPOSE  Perform the matrix-vector operation.
***LIBRARY   SLATEC (BLAS)
***CATEGORY  D1B4
***TYPE      COMPLEX (SHPMV-S, DHPMV-D, CHPMV-C)
***KEYWORDS  LEVEL 2 BLAS, LINEAR ALGEBRA
***AUTHOR   Dongarra, J. J., (ANL)
             Du Croz, J., (NAG)
             Hammarling, S., (NAG)
             Hanson, R. J., (SNLA)
***DESCRIPTION
```

CHPMV performs the matrix-vector operation

$$y := \alpha A x + \beta y,$$

where alpha and beta are scalars, x and y are n element vectors and A is an n by n hermitian matrix, supplied in packed form.

Parameters
=====

UPLO - CHARACTER*1.
On entry, UPLO specifies whether the upper or lower triangular part of the matrix A is supplied in the packed array AP as follows:

UPLO = 'U' or 'u' The upper triangular part of A is supplied in AP.

UPLO = 'L' or 'l' The lower triangular part of A is supplied in AP.

Unchanged on exit.

N - INTEGER.
On entry, N specifies the order of the matrix A. N must be at least zero.
Unchanged on exit.

ALPHA - COMPLEX .
On entry, ALPHA specifies the scalar alpha.
Unchanged on exit.

AP - COMPLEX array of DIMENSION at least
((n*(n + 1))/2).
Before entry with UPLO = 'U' or 'u', the array AP must contain the upper triangular part of the hermitian matrix packed sequentially, column by column, so that AP(1) contains a(1, 1), AP(2) and AP(3) contain a(1, 2) and a(2, 2) respectively, and so on.
Before entry with UPLO = 'L' or 'l', the array AP must contain the lower triangular part of the hermitian matrix packed sequentially, column by column, so that AP(1) contains a(1, 1), AP(2) and AP(3) contain a(2, 1)

and $a(3, 1)$ respectively, and so on.
 Note that the imaginary parts of the diagonal elements need not be set and are assumed to be zero.
 Unchanged on exit.

- X - COMPLEX array of dimension at least
 $(1 + (n - 1) * \text{abs}(\text{INCX}))$.
 Before entry, the incremented array X must contain the n
 element vector x.
 Unchanged on exit.

- INCX - INTEGER.
 On entry, INCX specifies the increment for the elements of
 X. INCX must not be zero.
 Unchanged on exit.

- BETA - COMPLEX .
 On entry, BETA specifies the scalar beta. When BETA is
 supplied as zero then Y need not be set on input.
 Unchanged on exit.

- Y - COMPLEX array of dimension at least
 $(1 + (n - 1) * \text{abs}(\text{INCY}))$.
 Before entry, the incremented array Y must contain the n
 element vector y. On exit, Y is overwritten by the updated
 vector y.

- INCY - INTEGER.
 On entry, INCY specifies the increment for the elements of
 Y. INCY must not be zero.
 Unchanged on exit.

***REFERENCES Dongarra, J. J., Du Croz, J., Hammarling, S., and
 Hanson, R. J. An extended set of Fortran basic linear
 algebra subprograms. ACM TOMS, Vol. 14, No. 1,
 pp. 1-17, March 1988.

***ROUTINES CALLED LSAME, XERBLA

***REVISION HISTORY (YYMMDD)

861022 DATE WRITTEN

910605 Modified to meet SLATEC prologue standards. Only comment
 lines were modified. (BKS)

END PROLOGUE

CHPR

```
SUBROUTINE CHPR (UPLO, N, ALPHA, X, INCX, AP)
***BEGIN PROLOGUE  CHPR
***PURPOSE  Perform the hermitian rank 1 operation.
***LIBRARY   SLATEC (BLAS)
***CATEGORY  D1B4
***TYPE      COMPLEX (CHPR-C)
***KEYWORDS  LEVEL 2 BLAS, LINEAR ALGEBRA
***AUTHOR   Dongarra, J. J., (ANL)
            Du Croz, J., (NAG)
            Hammarling, S., (NAG)
            Hanson, R. J., (SNLA)
***DESCRIPTION
```

CHPR performs the hermitian rank 1 operation

$$A := \alpha x \text{conjg}(x') + A,$$

where α is a real scalar, x is an n element vector and A is an n by n hermitian matrix, supplied in packed form.

Parameters
=====

UPLO - CHARACTER*1.
On entry, UPLO specifies whether the upper or lower triangular part of the matrix A is supplied in the packed array AP as follows:

UPLO = 'U' or 'u'	The upper triangular part of A is supplied in AP .
UPLO = 'L' or 'l'	The lower triangular part of A is supplied in AP .

Unchanged on exit.

N - INTEGER.
On entry, N specifies the order of the matrix A .
 N must be at least zero.
Unchanged on exit.

ALPHA - REAL.
On entry, $ALPHA$ specifies the scalar α .
Unchanged on exit.

X - COMPLEX array of dimension at least
(1 + (n - 1) * abs(INCX)).
Before entry, the incremented array X must contain the n element vector x .
Unchanged on exit.

INCX - INTEGER.
On entry, $INCX$ specifies the increment for the elements of X . $INCX$ must not be zero.
Unchanged on exit.

AP - COMPLEX array of DIMENSION at least
 ((n*(n + 1))/2).
 Before entry with UPLO = 'U' or 'u', the array AP must
 contain the upper triangular part of the hermitian matrix
 packed sequentially, column by column, so that AP(1)
 contains a(1, 1), AP(2) and AP(3) contain a(1, 2)
 and a(2, 2) respectively, and so on. On exit, the array
 AP is overwritten by the upper triangular part of the
 updated matrix.
 Before entry with UPLO = 'L' or 'l', the array AP must
 contain the lower triangular part of the hermitian matrix
 packed sequentially, column by column, so that AP(1)
 contains a(1, 1), AP(2) and AP(3) contain a(2, 1)
 and a(3, 1) respectively, and so on. On exit, the array
 AP is overwritten by the lower triangular part of the
 updated matrix.
 Note that the imaginary parts of the diagonal elements need
 not be set, they are assumed to be zero, and on exit they
 are set to zero.

***REFERENCES Dongarra, J. J., Du Croz, J., Hammarling, S., and
 Hanson, R. J. An extended set of Fortran basic linear
 algebra subprograms. ACM TOMS, Vol. 14, No. 1,
 pp. 1-17, March 1988.

***ROUTINES CALLED LSAME, XERBLA

***REVISION HISTORY (YYMMDD)
 861022 DATE WRITTEN
 910605 Modified to meet SLATEC prologue standards. Only comment
 lines were modified. (BKS)

END PROLOGUE

CHPR2

```
      SUBROUTINE CHPR2 (UPLO, N, ALPHA, X, INCX, Y, INCY, AP)
***BEGIN PROLOGUE  CHPR2
***PURPOSE  Perform the hermitian rank 2 operation.
***LIBRARY   SLATEC (BLAS)
***CATEGORY  D1B4
***TYPE      COMPLEX (SHPR2-S, DHPR2-D, CHPR2-C)
***KEYWORDS  LEVEL 2 BLAS, LINEAR ALGEBRA
***AUTHOR   Dongarra, J. J., (ANL)
            Du Croz, J., (NAG)
            Hammarling, S., (NAG)
            Hanson, R. J., (SNLA)
***DESCRIPTION
```

CHPR2 performs the hermitian rank 2 operation

$$A := \alpha x \text{conjg}(y') + \text{conjg}(\alpha) y \text{conjg}(x') + A,$$

where α is a scalar, x and y are n element vectors and A is an n by n hermitian matrix, supplied in packed form.

Parameters
=====

UPLO - CHARACTER*1.
On entry, UPLO specifies whether the upper or lower triangular part of the matrix A is supplied in the packed array AP as follows:

UPLO = 'U' or 'u'	The upper triangular part of A is supplied in AP .
UPLO = 'L' or 'l'	The lower triangular part of A is supplied in AP .

Unchanged on exit.

N - INTEGER.
On entry, N specifies the order of the matrix A .
 N must be at least zero.
Unchanged on exit.

ALPHA - COMPLEX .
On entry, ALPHA specifies the scalar α .
Unchanged on exit.

X - COMPLEX array of dimension at least
(1 + (n - 1) * abs(INCX)).
Before entry, the incremented array X must contain the n element vector x .
Unchanged on exit.

INCX - INTEGER.
On entry, INCX specifies the increment for the elements of X . INCX must not be zero.
Unchanged on exit.

Y - COMPLEX array of dimension at least
 (1 + (n - 1) * abs(INCY)).
 Before entry, the incremented array Y must contain the n
 element vector y.
 Unchanged on exit.

INCY - INTEGER.
 On entry, INCY specifies the increment for the elements of
 Y. INCY must not be zero.
 Unchanged on exit.

AP - COMPLEX array of DIMENSION at least
 ((n * (n + 1)) / 2).
 Before entry with UPLO = 'U' or 'u', the array AP must
 contain the upper triangular part of the hermitian matrix
 packed sequentially, column by column, so that AP(1)
 contains a(1, 1), AP(2) and AP(3) contain a(1, 2)
 and a(2, 2) respectively, and so on. On exit, the array
 AP is overwritten by the upper triangular part of the
 updated matrix.
 Before entry with UPLO = 'L' or 'l', the array AP must
 contain the lower triangular part of the hermitian matrix
 packed sequentially, column by column, so that AP(1)
 contains a(1, 1), AP(2) and AP(3) contain a(2, 1)
 and a(3, 1) respectively, and so on. On exit, the array
 AP is overwritten by the lower triangular part of the
 updated matrix.
 Note that the imaginary parts of the diagonal elements need
 not be set, they are assumed to be zero, and on exit they
 are set to zero.

***REFERENCES Dongarra, J. J., Du Croz, J., Hammarling, S., and
 Hanson, R. J. An extended set of Fortran basic linear
 algebra subprograms. ACM TOMS, Vol. 14, No. 1,
 pp. 1-17, March 1988.

***ROUTINES CALLED LSAME, XERBLA

***REVISION HISTORY (YYMMDD)
 861022 DATE WRITTEN
 910605 Modified to meet SLATEC prologue standards. Only comment
 lines were modified. (BKS)
 END PROLOGUE

CHPSL

```
SUBROUTINE CHPSL (AP, N, KPVT, B)
***BEGIN PROLOGUE  CHPSL
***PURPOSE  Solve a complex Hermitian system using factors obtained
            from CHPFA.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2D1A
***TYPE      COMPLEX (SSPSL-S, DSPSL-D, CHPSL-C, CSPSL-C)
***KEYWORDS  HERMITIAN, LINEAR ALGEBRA, LINPACK, MATRIX, PACKED, SOLVE
***AUTHOR   Bunch, J., (UCSD)
***DESCRIPTION
```

CHISL solves the complex Hermitian system
 $A * X = B$
using the factors computed by CHPFA.

On Entry

AP COMPLEX(N*(N+1)/2)
 the output from CHPFA.

N INTEGER
 the order of the matrix A .

KVPT INTEGER(N)
 the pivot vector from CHPFA.

B COMPLEX(N)
 the right hand side vector.

On Return

B the solution vector X .

Error Condition

A division by zero may occur if CHPCO has set RCOND .EQ. 0.0
or CHPFA has set INFO .NE. 0 .

To compute INVERSE(A) * C where C is a matrix
with P columns

```
CALL CHPFA(AP,N,KVPT,INFO)
IF (INFO .NE. 0) GO TO ...
DO 10 J = 1, P
    CALL CHPSL(AP,N,KVPT,C(1,J))
10 CONTINUE
```

```
***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
               Stewart, LINPACK Users' Guide, SIAM, 1979.
```

```
***ROUTINES CALLED  CAXPY, CDOTC
```

```
***REVISION HISTORY  (YYMMDD)
```

```
780814  DATE WRITTEN
890531  Changed all specific intrinsics to generic.  (WRB)
890831  Modified array declarations.  (WRB)
891107  Modified routine equivalence list.  (WRB)
891107  REVISION DATE from Version 3.2
891214  Prologue converted to Version 4.0 format.  (BAB)
```

900326 Removed duplicate information from DESCRIPTION section.
(WRB)
920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

CHU

```
      FUNCTION CHU (A, B, X)
***BEGIN PROLOGUE  CHU
***PURPOSE  Compute the logarithmic confluent hypergeometric function.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C11
***TYPE      SINGLE PRECISION (CHU-S, DCHU-D)
***KEYWORDS  FNLIB, LOGARITHMIC CONFLUENT HYPERGEOMETRIC FUNCTION,
              SPECIAL FUNCTIONS
***AUTHOR   Fullerton, W., (LANL)
***DESCRIPTION
```

CHU computes the logarithmic confluent hypergeometric function,
U(A,B,X).

Input Parameters:

```
      A   real
      B   real
      X   real and positive
```

This routine is not valid when $1+A-B$ is close to zero if X is small.

```
***REFERENCES  (NONE)
***ROUTINES CALLED  EXPREL, GAMMA, GAMR, POCH, POCH1, R1MACH, R9CHU,
                  XERMSG
***REVISION HISTORY  (YYMMDD)
      770801  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890531  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
      900727  Added EXTERNAL statement.  (WRB)
      END PROLOGUE
```

CINVIT

```
      SUBROUTINE CINVIT (NM, N, AR, AI, WR, WI, SELECT, MM, M, ZR, ZI,  
+      IERR, RM1, RM2, RV1, RV2)  
***BEGIN PROLOGUE  CINVIT  
***PURPOSE  Compute the eigenvectors of a complex upper Hessenberg  
            associated with specified eigenvalues using inverse  
            iteration.  
***LIBRARY    SLATEC (EISPACK)  
***CATEGORY   D4C2B  
***TYPE       COMPLEX (INVIT-S, CINVIT-C)  
***KEYWORDS   EIGENVALUES, EIGENVECTORS, EISPACK  
***AUTHOR    Smith, B. T., et al.  
***DESCRIPTION
```

This subroutine is a translation of the ALGOL procedure CXINVIT
by Peters and Wilkinson.
HANDBOOK FOR AUTO. COMP. VOL.II-LINEAR ALGEBRA, 418-439(1971).

This subroutine finds those eigenvectors of A COMPLEX UPPER
Hessenberg matrix corresponding to specified eigenvalues,
using inverse iteration.

On INPUT

NM must be set to the row dimension of the two-dimensional
array parameters, AR, AI, ZR and ZI, as declared in the
calling program dimension statement. NM is an INTEGER
variable.

N is the order of the matrix A=(AR,AI). N is an INTEGER
variable. N must be less than or equal to NM.

AR and AI contain the real and imaginary parts, respectively,
of the complex upper Hessenberg matrix. AR and AI are
two-dimensional REAL arrays, dimensioned AR(NM,N)
and AI(NM,N).

WR and WI contain the real and imaginary parts, respectively,
of the eigenvalues of the matrix. The eigenvalues must be
stored in a manner identical to that of subroutine COMLR,
which recognizes possible splitting of the matrix. WR and
WI are one-dimensional REAL arrays, dimensioned WR(N) and
WI(N).

SELECT specifies the eigenvectors to be found. The
eigenvector corresponding to the J-th eigenvalue is
specified by setting SELECT(J) to .TRUE. SELECT is a
one-dimensional LOGICAL array, dimensioned SELECT(N).

MM should be set to an upper bound for the number of
eigenvectors to be found. MM is an INTEGER variable.

On OUTPUT

AR, AI, WI, and SELECT are unaltered.

WR may have been altered since close eigenvalues are perturbed

slightly in searching for independent eigenvectors.

M is the number of eigenvectors actually found. M is an INTEGER variable.

ZR and ZI contain the real and imaginary parts, respectively, of the eigenvectors corresponding to the flagged eigenvalues. The eigenvectors are normalized so that the component of largest magnitude is 1. Any vector which fails the acceptance test is set to zero. ZR and ZI are two-dimensional REAL arrays, dimensioned ZR(NM,MM) and ZI(NM,MM).

IERR is an INTEGER flag set to
Zero for normal return,
-(2*N+1) if more than MM eigenvectors have been requested
(the MM eigenvectors calculated to this point are in ZR and ZI),
-K if the iteration corresponding to the K-th value fails (if this occurs more than once, K is the index of the last occurrence); the corresponding columns of ZR and ZI are set to zero vectors,
-(N+K) if both error situations occur.

RV1 and RV2 are one-dimensional REAL arrays used for temporary storage, dimensioned RV1(N) and RV2(N). They hold the approximate eigenvectors during the inverse iteration process.

RM1 and RM2 are two-dimensional REAL arrays used for temporary storage, dimensioned RM1(N,N) and RM2(N,N). These arrays hold the triangularized form of the upper Hessenberg matrix used in the inverse iteration process.

The ALGOL procedure GUESSVEC appears in CINVIT in-line.

Calls PYTHAG(A,B) for $\sqrt{A^2 + B^2}$.
Calls CDIV for complex division.

Questions and comments should be directed to B. S. Garbow,
APPLIED MATHEMATICS DIVISION, ARGONNE NATIONAL LABORATORY

***REFERENCES B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow,
Y. Ikebe, V. C. Klema and C. B. Moler, Matrix Eigen-
system Routines - EISPACK Guide, Springer-Verlag,
1976.

***ROUTINES CALLED CDIV, PYTHAG

***REVISION HISTORY (YYMMDD)

760101 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CLBETA

```
      COMPLEX FUNCTION CLBETA (A, B)
***BEGIN PROLOGUE  CLBETA
***PURPOSE  Compute the natural logarithm of the complete Beta
            function.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C7B
***TYPE      COMPLEX (ALBETA-S, DLBETA-D, CLBETA-C)
***KEYWORDS  FNLIB, LOGARITHM OF THE COMPLETE BETA FUNCTION,
            SPECIAL FUNCTIONS
***AUTHOR   Fullerton, W., (LANL)
***DESCRIPTION
```

CLBETA computes the natural log of the complex valued complete beta function of complex parameters A and B. This is a preliminary version which is not accurate.

Input Parameters:

 A complex and the real part of A positive
 B complex and the real part of B positive

```
***REFERENCES  (NONE)
***ROUTINES CALLED  CLNGAM, XERMSG
***REVISION HISTORY  (YYMMDD)
    770701  DATE WRITTEN
    861211  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
    END PROLOGUE
```

CLNGAM

```
      COMPLEX FUNCTION CLNGAM (ZIN)
***BEGIN PROLOGUE  CLNGAM
***PURPOSE  Compute the logarithm of the absolute value of the Gamma
            function.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C7A
***TYPE      COMPLEX (ALNGAM-S, DLNGAM-D, CLNGAM-C)
***KEYWORDS  ABSOLUTE VALUE, COMPLETE GAMMA FUNCTION, FNLIB, LOGARITHM,
            SPECIAL FUNCTIONS
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION
```

CLNGAM computes the natural log of the complex valued gamma function at ZIN, where ZIN is a complex number. This is a preliminary version, which is not accurate.

```
***REFERENCES  (NONE)
***ROUTINES CALLED  C9LGMC, CARG, CLNREL, R1MACH, XERMSG
***REVISION HISTORY  (YMMDD)
    780401  DATE WRITTEN
    890531  Changed all specific intrinsics to generic.  (WRB)
    890531  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
END PROLOGUE
```

CLNREL

```
      COMPLEX FUNCTION CLNREL (Z)
***BEGIN PROLOGUE  CLNREL
***PURPOSE  Evaluate  $\ln(1+X)$  accurate in the sense of relative error.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C4B
***TYPE      COMPLEX (ALNREL-S, DLNREL-D, CLNREL-C)
***KEYWORDS  ELEMENTARY FUNCTIONS, FNLIB, LOGARITHM
***AUTHOR   Fullerton, W., (LANL)
***DESCRIPTION

      CLNREL(Z) = LOG(1+Z) with relative error accuracy near  $Z = 0$ .
      Let  $RHO = ABS(Z)$  and
            $R**2 = ABS(1+Z)**2 = (1+X)**2 + Y**2 = 1 + 2*X + RHO**2$  .
      Now if RHO is small we may evaluate CLNREL(Z) accurately by
           LOG(1+Z) = CMPLX (LOG(R), CARG(1+Z))
                    = CMPLX (0.5*LOG(R**2), CARG(1+Z))
                    = CMPLX (0.5*ALNREL(2*X+RHO**2), CARG(1+Z))

***REFERENCES  (NONE)
***ROUTINES CALLED  ALNREL, CARG, R1MACH, XERMSG
***REVISION HISTORY  (YYMMDD)
      770401  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890531  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
      END PROLOGUE
```

CLOG10

```
      COMPLEX FUNCTION CLOG10 (Z)
***BEGIN PROLOGUE  CLOG10
***PURPOSE  Compute the principal value of the complex base 10
            logarithm.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C4B
***TYPE      COMPLEX (CLOG10-C)
***KEYWORDS  BASE TEN LOGARITHM, ELEMENTARY FUNCTIONS, FNLIB
***AUTHOR   Fullerton, W., (LANL)
***DESCRIPTION

      CLOG10(Z) calculates the principal value of the complex common
      or base 10 logarithm of Z for  $-\pi \leq \arg(Z) \leq \pi$ .

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      770401  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890531  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      END PROLOGUE
```

CMGNBN

```

      SUBROUTINE CMGNBN (NPEROD, N, MPEROD, M, A, B, C, IDIMY, Y,
+      IERROR, W)
***BEGIN PROLOGUE  CMGNBN
***PURPOSE  Solve a complex block tridiagonal linear system of
              equations by a cyclic reduction algorithm.
***LIBRARY    SLATEC (FISHPACK)
***CATEGORY   I2B4B
***TYPE       COMPLEX (GENBUN-S, CMGNBN-C)
***KEYWORDS   CYCLIC REDUCTION, ELLIPTIC PDE, FISHPACK,
              TRIDIAGONAL LINEAR SYSTEM
***AUTHOR     Adams, J., (NCAR)
              Swarztrauber, P. N., (NCAR)
              Sweet, R., (NCAR)
***DESCRIPTION

```

Subroutine CMGNBN solves the complex linear system of equations

$$A(I)*X(I-1,J) + B(I)*X(I,J) + C(I)*X(I+1,J) \\ + X(I,J-1) - 2.*X(I,J) + X(I,J+1) = Y(I,J)$$

For $I = 1, 2, \dots, M$ and $J = 1, 2, \dots, N$.

The indices $I+1$ and $I-1$ are evaluated modulo M , i.e., $X(0,J) = X(M,J)$ and $X(M+1,J) = X(1,J)$, and $X(I,0)$ may be equal to 0, $X(I,2)$, or $X(I,N)$ and $X(I,N+1)$ may be equal to 0, $X(I,N-1)$, or $X(I,1)$ depending on an input parameter.

* * * * *	Parameter Description	* * * * *
* * * * *	On Input	* * * * *

NPEROD

Indicates the values that $X(I,0)$ and $X(I,N+1)$ are assumed to have.

- = 0 If $X(I,0) = X(I,N)$ and $X(I,N+1) = X(I,1)$.
- = 1 If $X(I,0) = X(I,N+1) = 0$.
- = 2 If $X(I,0) = 0$ and $X(I,N+1) = X(I,N-1)$.
- = 3 If $X(I,0) = X(I,2)$ and $X(I,N+1) = X(I,N-1)$.
- = 4 If $X(I,0) = X(I,2)$ and $X(I,N+1) = 0$.

N

The number of unknowns in the J-direction. N must be greater than 2.

MPEROD

- = 0 If $A(1)$ and $C(M)$ are not zero
- = 1 If $A(1) = C(M) = 0$

M

The number of unknowns in the I-direction. N must be greater than 2.

A,B,C

One-dimensional complex arrays of length M that specify the coefficients in the linear equations given above. If MPEROD = 0 the array elements must not depend upon the index I, but must be constant. Specifically, the subroutine checks the following condition

$$\begin{aligned} A(I) &= C(1) \\ C(I) &= C(1) \\ B(I) &= B(1) \end{aligned}$$

For $I=1,2,\dots,M$.

IDIMY

The row (or first) dimension of the two-dimensional array Y as it appears in the program calling CMGNBN. This parameter is used to specify the variable dimension of Y. IDIMY must be at least M.

Y

A two-dimensional complex array that specifies the values of the right side of the linear system of equations given above. Y must be dimensioned at least M*N.

W

A one-dimensional complex array that must be provided by the user for work space. W may require up to $4*N + (10 + \text{INT}(\log_2(N)))*M$ LOCATIONS. The actual number of locations used is computed by CMGNBN and is returned in location W(1).

* * * * * On Output * * * * *

Y

Contains the solution X.

IERROR

An error flag which indicates invalid input parameters. Except for number zero, a solution is not attempted.

= 0 No error.
 = 1 M .LE. 2
 = 2 N .LE. 2
 = 3 IDIMY .LT. M
 = 4 NPEROD .LT. 0 or NPEROD .GT. 4
 = 5 MPEROD .LT. 0 or MPEROD .GT. 1
 = 6 A(I) .NE. C(1) or C(I) .NE. C(1) or B(I) .NE. B(1) for some $I=1,2,\dots,M$.
 = 7 A(1) .NE. 0 or C(M) .NE. 0 and MPEROD = 1

W

W(1) contains the required length of W.

*Long Description:

* * * * * Program Specifications * * * * *

Dimension of Arguments A(M),B(M),C(M),Y(IDIMY,N),W(see parameter list)

Latest June 1979

SLATEC2 (AAAAAA through D9UPAK) - 303

Revision

Subprograms Required	CMGNBN,CMPOSD,CMPOSN,CMPOSP,CMPCSG,CMPMRG,CMPTRX,CMPTR3,PIMACH
Special Conditions	None
Common Blocks	None
I/O	None
Precision	Single
Specialist	Roland Sweet
Language	FORTTRAN
History	Written by Roland Sweet at NCAR in June, 1977
Algorithm	The linear system is solved by a cyclic reduction algorithm described in the reference.
Space Required	4944(DECIMAL) = 11520(octal) locations on the NCAR Control Data 7600

Timing and Accuracy The execution time T on the NCAR Control Data 7600 for subroutine CMGNBN is roughly proportional to $M*N*\log_2(N)$, but also depends on the input parameter NPEROD. Some typical values are listed in the table below.

To measure the accuracy of the algorithm a uniform random number generator was used to create a solution array X for the system given in the 'PURPOSE' with

$$A(I) = C(I) = -0.5*B(I) = 1, \quad I=1,2,\dots,M$$

and, when MPEROD = 1

$$\begin{aligned} A(1) &= C(M) = 0 \\ A(M) &= C(1) = 2. \end{aligned}$$

The solution X was substituted into the given system and a right side Y was computed. Using this array Y subroutine CMGNBN was called to produce an approximate solution Z. Then the relative error, defined as

$$E = \text{MAX}(\text{ABS}(Z(I,J)-X(I,J)))/\text{MAX}(\text{ABS}(X(I,J)))$$

where the two maxima are taken over all $I=1,2,\dots,M$ and $J=1,2,\dots,N$, was computed. The value of E is given in the table below for some typical values of M and N.

M (=N)	MPEROD	NPEROD	T(MSECS)	E
-----	-----	-----	-----	-----

31	0	0	77	1.E-12
31	1	1	45	4.E-13
31	1	3	91	2.E-12
32	0	0	59	7.E-14
32	1	1	65	5.E-13
32	1	3	97	2.E-13
33	0	0	80	6.E-13
33	1	1	67	5.E-13
33	1	3	76	3.E-12
63	0	0	350	5.E-12
63	1	1	215	6.E-13
63	1	3	412	1.E-11
64	0	0	264	1.E-13
64	1	1	287	3.E-12
64	1	3	421	3.E-13
65	0	0	338	2.E-12
65	1	1	292	5.E-13
65	1	3	329	1.E-11

Portability American National Standards Institute Fortran.
The machine dependent constant PI is defined in
function PIMACH.

Required COS
Resident
Routines

Reference Sweet, R., 'A Cyclic Reduction Algorithm for
Solving Block Tridiagonal Systems Of Arbitrary
Dimensions,' SIAM J. on Numer. Anal.,
14(SEPT., 1977), PP. 706-720.

* * * * *

***REFERENCES R. Sweet, A cyclic reduction algorithm for solving
 block tridiagonal systems of arbitrary dimensions,
 SIAM Journal on Numerical Analysis 14, (September
 1977), pp. 706-720.

***ROUTINES CALLED CMPOSD, CMPOSN, CMPOSP

***REVISION HISTORY (YYMMDD)

801001 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890531 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CNBCO

```
SUBROUTINE CNBCO (ABE, LDA, N, ML, MU, IPVT, RCOND, Z)
***BEGIN PROLOGUE  CNBCO
***PURPOSE  Factor a band matrix using Gaussian elimination and
            estimate the condition number.
***LIBRARY   SLATEC
***CATEGORY  D2C2
***TYPE      COMPLEX (SNBCO-S, DNBCO-D, CNBCO-C)
***KEYWORDS  BANDED, LINEAR EQUATIONS, MATRIX FACTORIZATION,
            NONSYMMETRIC
***AUTHOR  Voorhees, E. A., (LANL)
***DESCRIPTION
```

CNBCO factors a complex band matrix by Gaussian elimination and estimates the condition of the matrix.

If RCOND is not needed, CNBFA is slightly faster.
To solve $A \cdot X = B$, follow CNBCO by CNBSL.
To compute $\text{INVERSE}(A) \cdot C$, follow CNBCO by CNBSL.
To compute $\text{DETERMINANT}(A)$, follow CNBCO by CNBDI.

On Entry

ABE COMPLEX(LDA, NC)
 contains the matrix in band storage. The rows
 of the original matrix are stored in the rows
 of ABE and the diagonals of the original matrix
 are stored in columns 1 through ML+MU+1 of ABE.
 NC must be .GE. 2*ML+MU+1 .
 See the comments below for details.

LDA INTEGER
 the leading dimension of the array ABE.
 LDA must be .GE. N .

N INTEGER
 the order of the original matrix.

ML INTEGER
 number of diagonals below the main diagonal.
 0 .LE. ML .LT. N .

MU INTEGER
 number of diagonals above the main diagonal.
 0 .LE. MU .LT. N .
 More efficient if ML .LE. MU .

On Return

ABE an upper triangular matrix in band storage
 and the multipliers which were used to obtain it.
 The factorization can be written $A = L \cdot U$ where
 L is a product of permutation and unit lower
 triangular matrices and U is upper triangular.

IPVT INTEGER(N)
 an integer vector of pivot indices.

RCOND REAL
 an estimate of the reciprocal condition of A .
 For the system $A \cdot X = B$, relative perturbations
 in A and B of size EPSILON may cause
 relative perturbations in X of size $\text{EPSILON}/\text{RCOND}$.
 If RCOND is so small that the logical expression
 $1.0 + \text{RCOND} \text{ .EQ. } 1.0$
 is true, then A may be singular to working
 precision. In particular, RCOND is zero if
 exact singularity is detected or the estimate
 underflows.

Z COMPLEX(N)
 a work vector whose contents are usually unimportant.
 If A is close to a singular matrix, then Z is
 an approximate null vector in the sense that
 $\text{NORM}(A \cdot Z) = \text{RCOND} \cdot \text{NORM}(A) \cdot \text{NORM}(Z)$.

Band Storage

If A is a band matrix, the following program segment
 will set up the input.

```

      ML = (band width below the diagonal)
      MU = (band width above the diagonal)
      DO 20 I = 1, N
        J1 = MAX(1, I-ML)
        J2 = MIN(N, I+MU)
        DO 10 J = J1, J2
          K = J - I + ML + 1
          ABE(I,K) = A(I,J)
        10 CONTINUE
      20 CONTINUE
```

This uses columns 1 through ML+MU+1 of ABE .
 Furthermore, ML additional columns are needed in
 ABE starting with column ML+MU+2 for elements
 generated during the triangularization. The total
 number of columns needed in ABE is $2 \cdot \text{ML} + \text{MU} + 1$.

Example: If the original matrix is

```

11 12 13  0  0  0
21 22 23 24  0  0
 0 32 33 34 35  0
 0  0 43 44 45 46
 0  0  0 54 55 56
 0  0  0  0 65 66
```

then $N = 6$, $\text{ML} = 1$, $\text{MU} = 2$, $\text{LDA} \geq 5$ and ABE should contain

```

  * 11 12 13  +      , * = not used
21 22 23 24  +      , + = used for pivoting
32 33 34 35  +
43 44 45 46  +
54 55 56  *  +
65 66  *  *  +
```

Stewart, LINPACK Users' Guide, SIAM, 1979.
***ROUTINES CALLED CAXPY, CDOTC, CNBFA, CSSCAL, SCASUM
***REVISION HISTORY (YYMMDD)
800730 DATE WRITTEN
890531 Changed all specific intrinsics to generic. (WRB)
890831 Modified array declarations. (WRB)
890831 REVISION DATE from Version 3.2
891214 Prologue converted to Version 4.0 format. (BAB)
920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

CNBDI

```
SUBROUTINE CNBDI (ABE, LDA, N, ML, MU, IPVT, DET)
***BEGIN PROLOGUE  CNBDI
***PURPOSE  Compute the determinant of a band matrix using the factors
             computed by CNBCO or CNBFA.
***LIBRARY   SLATEC
***CATEGORY  D3C2
***TYPE      COMPLEX (SNBDI-S, DNBBDI-D, CNBDI-C)
***KEYWORDS  BANDED, DETERMINANT, LINEAR EQUATIONS, NONSYMMETRIC
***AUTHOR  Voorhees, E. A., (LANL)
***DESCRIPTION
```

CNBDI computes the determinant of a band matrix
using the factors computed by CNBCO or CNBFA.
If the inverse is needed, use CNBSL N times.

On Entry

ABE COMPLEX(LDA, NC)
 the output from CNBCO or CNBFA.
 NC must be .GE. 2*ML+MU+1 .

LDA INTEGER
 the leading dimension of the array ABE .

N INTEGER
 the order of the original matrix.

ML INTEGER
 number of diagonals below the main diagonal.

MU INTEGER
 number of diagonals above the main diagonal.

IPVT INTEGER(N)
 the pivot vector from CNBCO or CNBFA.

On Return

DET COMPLEX(2)
 determinant of original matrix.
 Determinant = DET(1) * 10.0**DET(2)
 with 1.0 .LE. CABS1(DET(1)) .LT. 10.0
 or DET(1) = 0.0 .

```
***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
               Stewart, LINPACK Users' Guide, SIAM, 1979.
```

```
***ROUTINES CALLED  (NONE)
```

```
***REVISION HISTORY  (YMMDD)
```

```
800730  DATE WRITTEN
890831  Modified array declarations.  (WRB)
890831  REVISION DATE from Version 3.2
891214  Prologue converted to Version 4.0 format.  (BAB)
920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

CNBFA

```
      SUBROUTINE CNBFA (ABE, LDA, N, ML, MU, IPVT, INFO)
***BEGIN PROLOGUE  CNBFA
***PURPOSE  Factor a band matrix by elimination.
***LIBRARY   SLATEC
***CATEGORY  D2C2
***TYPE      COMPLEX (SNBFA-S, DNBFA-D, CNBFA-C)
***KEYWORDS  BANDED, LINEAR EQUATIONS, MATRIX FACTORIZATION,
              NONSYMMETRIC
***AUTHOR   Voorhees, E. A., (LANL)
***DESCRIPTION
```

CNBFA factors a complex band matrix by elimination.

CNBFA is usually called by CNBCO, but it can be called directly with a saving in time if RCOND is not needed.

On Entry

ABE COMPLEX(LDA, NC)
 contains the matrix in band storage. The rows
 of the original matrix are stored in the rows
 of ABE and the diagonals of the original matrix
 are stored in columns 1 through ML+MU+1 of ABE.
 NC must be .GE. 2*ML+MU+1 .
 See the comments below for details.

LDA INTEGER
 the leading dimension of the array ABE.
 LDA must be .GE. N .

N INTEGER
 the order of the original matrix.

ML INTEGER
 number of diagonals below the main diagonal.
 0 .LE. ML .LT. N .

MU INTEGER
 number of diagonals above the main diagonal.
 0 .LE. MU .LT. N .
 More efficient if ML .LE. MU .

On Return

ABE an upper triangular matrix in band storage
 and the multipliers which were used to obtain it.
 the factorization can be written $A = L*U$ where
 L is a product of permutation and unit lower
 triangular matrices and U is upper triangular.

IPVT INTEGER(N)
 an integer vector of pivot indices.

INFO INTEGER
 =0 normal value
 =K if U(K,K) .EQ. 0.0 . This is not an error

condition for this subroutine, but it does indicate that CNBSL will divide by zero if called. Use RCOND in CNBCO for a reliable indication of singularity.

Band Storage

If A is a band matrix, the following program segment will set up the input.

```

      ML = (band width below the diagonal)
      MU = (band width above the diagonal)
      DO 20 I = 1, N
        J1 = MAX(1, I-ML)
        J2 = MIN(N, I+MU)
        DO 10 J = J1, J2
          K = J - I + ML + 1
          ABE(I,K) = A(I,J)
        10 CONTINUE
      20 CONTINUE

```

This uses columns 1 through ML+MU+1 of ABE . Furthermore, ML additional columns are needed in ABE starting with column ML+MU+2 for elements generated during the triangularization. The total number of columns needed in ABE is 2*ML+MU+1 .

Example: If the original matrix is

```

11 12 13  0  0  0
21 22 23 24  0  0
 0 32 33 34 35  0
 0  0 43 44 45 46
 0  0  0 54 55 56
 0  0  0  0 65 66

```

then N = 6, ML = 1, MU = 2, LDA .GE. 5 and ABE should contain

```

* 11 12 13  +      , * = not used
21 22 23 24  +      , + = used for pivoting
32 33 34 35  +
43 44 45 46  +
54 55 56  *  +
65 66  *  *  +

```

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CAXPY, CSCAL, CSWAP, ICAMAX

***REVISION HISTORY (YMMDD)

800730 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CNBFS

```
      SUBROUTINE CNBFS (ABE, LDA, N, ML, MU, V, ITASK, IND, WORK, IWORK)
***BEGIN PROLOGUE  CNBFS
***PURPOSE  Solve a general nonsymmetric banded system of linear
            equations.
***LIBRARY   SLATEC
***CATEGORY  D2C2
***TYPE      COMPLEX (SNBFS-S, DNBFS-D, CNBFS-C)
***KEYWORDS  BANDED, LINEAR EQUATIONS, NONSYMMETRIC
***AUTHOR   Voorhees, E. A., (LANL)
***DESCRIPTION
```

Subroutine CNBFS solves a general nonsymmetric banded NxN system of single precision complex linear equations using SLATEC subroutines CNBCO and CNBSL. These are adaptations of the LINPACK subroutines CGBCO and CGBSL which require a different format for storing the matrix elements. If A is an NxN complex matrix and if X and B are complex N-vectors, then CNBFS solves the equation

$$A \cdot X = B.$$

A band matrix is a matrix whose nonzero elements are all fairly near the main diagonal, specifically $A(I,J) = 0$ if $I-J$ is greater than ML or $J-I$ is greater than MU. The integers ML and MU are called the lower and upper band widths and $M = ML+MU+1$ is the total band width. CNBFS uses less time and storage than the corresponding program for general matrices (CGEFS) if $2 \cdot ML + MU \leq N$.

The matrix A is first factored into upper and lower triangular matrices U and L using partial pivoting. These factors and the pivoting information are used to find the solution vector X. An approximate condition number is calculated to provide a rough estimate of the number of digits of accuracy in the computed solution.

If the equation $A \cdot X = B$ is to be solved for more than one vector B, the factoring of A does not need to be performed again and the option to only solve (ITASK .GT. 1) will be faster for the succeeding solutions. In this case, the contents of A, LDA, N and IWORK must not have been altered by the user following factorization (ITASK=1). IND will not be changed by CNBFS in this case.

Band Storage

If A is a band matrix, the following program segment will set up the input.

```
      ML = (band width below the diagonal)
      MU = (band width above the diagonal)
      DO 20 I = 1, N
          J1 = MAX(1, I-ML)
          J2 = MIN(N, I+MU)
          DO 10 J = J1, J2
```

```

          K = J - I + ML + 1
          ABE(I,K) = A(I,J)
10      CONTINUE
20 CONTINUE

```

This uses columns 1 through ML+MU+1 of ABE .
 Furthermore, ML additional columns are needed in
 ABE starting with column ML+MU+2 for elements
 generated during the triangularization. The total
 number of columns needed in ABE is 2*ML+MU+1 .

Example: If the original matrix is

```

11 12 13 0 0 0
21 22 23 24 0 0
0 32 33 34 35 0
0 0 43 44 45 46
0 0 0 54 55 56
0 0 0 0 65 66

```

then N = 6, ML = 1, MU = 2, LDA .GE. 5 and ABE should contain

```

* 11 12 13 +      , * = not used
21 22 23 24 +      , + = used for pivoting
32 33 34 35 +
43 44 45 46 +
54 55 56 * +
65 66 * * +

```

Argument Description ***

ABE COMPLEX(LDA,NC)
 on entry, contains the matrix in band storage as
 described above. NC must not be less than
 2*ML+MU+1 . The user is cautioned to specify NC
 with care since it is not an argument and cannot
 be checked by CNBFS. The rows of the original
 matrix are stored in the rows of ABE and the
 diagonals of the original matrix are stored in
 columns 1 through ML+MU+1 of ABE .
 on return, contains an upper triangular matrix U and
 the multipliers necessary to construct a matrix L
 so that A=L*U.

LDA INTEGER
 the leading dimension of array ABE. LDA must be great-
 er than or equal to N. (terminal error message IND=-1)

N INTEGER
 the order of the matrix A. N must be greater
 than or equal to 1 . (terminal error message IND=-2)

ML INTEGER
 the number of diagonals below the main diagonal.
 ML must not be less than zero nor greater than or
 equal to N . (terminal error message IND=-5)

MU INTEGER
 the number of diagonals above the main diagonal.
 MU must not be less than zero nor greater than or
 equal to N . (terminal error message IND=-6)

V COMPLEX(N)
 on entry, the singly subscripted array(vector) of di-

mension N which contains the right hand side B of a
 system of simultaneous linear equations $A \cdot X = B$.
 on return, V contains the solution vector, X .
 ITASK INTEGER
 if ITASK = 1, the matrix A is factored and then the
 linear equation is solved.
 if ITASK .GT. 1, the equation is solved using the existing
 factored matrix A and IWORK.
 if ITASK .LT. 1, then terminal error message IND=-3 is
 printed.
 IND INTEGER
 GT. 0 IND is a rough estimate of the number of digits
 of accuracy in the solution, X.
 LT. 0 see error message corresponding to IND below.
 WORK COMPLEX(N)
 a singly subscripted array of dimension at least N.
 IWORK INTEGER(N)
 a singly subscripted array of dimension at least N.

Error Messages Printed ***

IND=-1	terminal	N is greater than LDA.
IND=-2	terminal	N is less than 1.
IND=-3	terminal	ITASK is less than 1.
IND=-4	terminal	The matrix A is computationally singular. A solution has not been computed.
IND=-5	terminal	ML is less than zero or is greater than or equal to N .
IND=-6	terminal	MU is less than zero or is greater than or equal to N .
IND=-10	warning	The solution has no apparent significance. The solution may be inaccurate or the matrix A may be poorly scaled.

NOTE- The above terminal(*fatal*) error messages are
 designed to be handled by XERMSG in which
 LEVEL=1 (recoverable) and IFLAG=2 . LEVEL=0
 for warning error messages from XERMSG. Unless
 the user provides otherwise, an error message
 will be printed followed by an abort.

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
 Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CNBCO, CNBSL, R1MACH, XERMSG

***REVISION HISTORY (YYMMDD)

800813 DATE WRITTEN
 890531 Changed all specific intrinsics to generic. (WRB)
 890831 Modified array declarations. (WRB)
 890831 REVISION DATE from Version 3.2
 891214 Prologue converted to Version 4.0 format. (BAB)
 900315 CALLs to XERROR changed to CALLs to XERMSG. (THJ)
 900510 Convert XERRWV calls to XERMSG calls, cvt GOTO's to
 IF-THEN-ELSE. (RWC)
 920501 Reformatted the REFERENCES section. (WRB)
 END PROLOGUE

CNBIR

```
SUBROUTINE CNBIR (ABE, LDA, N, ML, MU, V, ITASK, IND, WORK, IWORK)
***BEGIN PROLOGUE  CNBIR
***PURPOSE  Solve a general nonsymmetric banded system of linear
            equations.  Iterative refinement is used to obtain an error
            estimate.
***LIBRARY    SLATEC
***CATEGORY  D2C2
***TYPE       COMPLEX (SNBIR-S, CNBIR-C)
***KEYWORDS   BANDED, LINEAR EQUATIONS, NONSYMMETRIC
***AUTHOR    Voorhees, E. A., (LANL)
***DESCRIPTION
```

Subroutine CNBIR solves a general nonsymmetric banded $N \times N$ system of single precision complex linear equations using SLATEC subroutines CNBFA and CNBSL. These are adaptations of the LINPACK subroutines CGBFA and CGBSL which require a different format for storing the matrix elements. One pass of iterative refinement is used only to obtain an estimate of the accuracy. If A is an $N \times N$ complex banded matrix and if X and B are complex N -vectors, then CNBIR solves the equation

$$A \cdot X = B.$$

A band matrix is a matrix whose nonzero elements are all fairly near the main diagonal, specifically $A(I,J) = 0$ if $I-J$ is greater than ML or $J-I$ is greater than MU . The integers ML and MU are called the lower and upper band widths and $M = ML + MU + 1$ is the total band width. CNBIR uses less time and storage than the corresponding program for general matrices (CGEIR) if $2 \cdot ML + MU \leq N$.

The matrix A is first factored into upper and lower triangular matrices U and L using partial pivoting. These factors and the pivoting information are used to find the solution vector X . Then the residual vector is found and used to calculate an estimate of the relative error, IND . IND estimates the accuracy of the solution only when the input matrix and the right hand side are represented exactly in the computer and does not take into account any errors in the input data.

If the equation $A \cdot X = B$ is to be solved for more than one vector B , the factoring of A does not need to be performed again and the option to only solve ($ITASK \geq 1$) will be faster for the succeeding solutions. In this case, the contents of A , LDA , N , $WORK$ and $IWORK$ must not have been altered by the user following factorization ($ITASK=1$). IND will not be changed by CNBIR in this case.

Band Storage

If A is a band matrix, the following program segment will set up the input.

$ML = (\text{band width below the diagonal})$

```

        MU = (band width above the diagonal)
DO 20 I = 1, N
    J1 = MAX(1, I-ML)
    J2 = MIN(N, I+MU)
    DO 10 J = J1, J2
        K = J - I + ML + 1
        ABE(I,K) = A(I,J)
    10 CONTINUE
20 CONTINUE

```

This uses columns 1 through ML+MU+1 of ABE .

Example: If the original matrix is

```

11 12 13  0  0  0
21 22 23 24  0  0
 0 32 33 34 35  0
 0  0 43 44 45 46
 0  0  0 54 55 56
 0  0  0  0 65 66

```

then N = 6, ML = 1, MU = 2, LDA .GE. 5 and ABE should contain

```

* 11 12 13          , * = not used
21 22 23 24
32 33 34 35
43 44 45 46
54 55 56 *
65 66 * *

```

Argument Description ***

ABE	<p>COMPLEX(LDA,MM)</p> <p>on entry, contains the matrix in band storage as described above. MM must not be less than M = ML+MU+1 . The user is cautioned to dimension ABE with care since MM is not an argument and cannot be checked by CNBIR. The rows of the original matrix are stored in the rows of ABE and the diagonals of the original matrix are stored in columns 1 through ML+MU+1 of ABE . ABE is not altered by the program.</p>
LDA	<p>INTEGER</p> <p>the leading dimension of array ABE. LDA must be greater than or equal to N. (terminal error message IND=-1)</p>
N	<p>INTEGER</p> <p>the order of the matrix A. N must be greater than or equal to 1 . (terminal error message IND=-2)</p>
ML	<p>INTEGER</p> <p>the number of diagonals below the main diagonal. ML must not be less than zero nor greater than or equal to N . (terminal error message IND=-5)</p>
MU	<p>INTEGER</p> <p>the number of diagonals above the main diagonal. MU must not be less than zero nor greater than or equal to N . (terminal error message IND=-6)</p>
V	<p>COMPLEX(N)</p> <p>on entry, the singly subscripted array(vector) of dimension N which contains the right hand side B of a</p>

system of simultaneous linear equations $A \cdot X = B$.
on return, V contains the solution vector, X .

ITASK INTEGER
if ITASK=1, the matrix A is factored and then the
linear equation is solved.
if ITASK .GT. 1, the equation is solved using the existing
factored matrix A and IWORK.
if ITASK .LT. 1, then terminal error message IND=-3 is
printed.

IND INTEGER
GT. 0 IND is a rough estimate of the number of digits
of accuracy in the solution, X . IND=75 means
that the solution vector X is zero.
LT. 0 see error message corresponding to IND below.

WORK COMPLEX(N*(NC+1))
a singly subscripted array of dimension at least
N*(NC+1) where NC = 2*ML+MU+1 .

IWORK INTEGER(N)
a singly subscripted array of dimension at least N.

Error Messages Printed ***

IND=-1	terminal	N is greater than LDA.
IND=-2	terminal	N is less than 1.
IND=-3	terminal	ITASK is less than 1.
IND=-4	terminal	The matrix A is computationally singular. A solution has not been computed.
IND=-5	terminal	ML is less than zero or is greater than or equal to N .
IND=-6	terminal	MU is less than zero or is greater than or equal to N .
IND=-10	warning	The solution has no apparent significance. The solution may be inaccurate or the matrix A may be poorly scaled.

NOTE- The above terminal(*fatal*) error messages are
designed to be handled by XERMSG in which
LEVEL=1 (recoverable) and IFLAG=2 . LEVEL=0
for warning error messages from XERMSG. Unless
the user provides otherwise, an error message
will be printed followed by an abort.

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CCOPY, CDCDOT, CNBFA, CNBSL, R1MACH, SCASUM, XERMSG

***REVISION HISTORY (YYMMDD)

800819 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900315 CALLs to XERROR changed to CALLs to XERMSG. (THJ)

900510 Convert XERRWV calls to XERMSG calls, cvt GOTO's to
IF-THEN-ELSE. (RWC)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CNBSL

```
SUBROUTINE CNBSL (ABE, LDA, N, ML, MU, IPVT, B, JOB)
***BEGIN PROLOGUE  CNBSL
***PURPOSE  Solve a complex band system using the factors computed by
             CNBCO or CNBFA.
***LIBRARY   SLATEC
***CATEGORY  D2C2
***TYPE      COMPLEX (SNBSL-S, DNBSL-D, CNBSL-C)
***KEYWORDS  BANDED, LINEAR EQUATIONS, NONSYMMETRIC, SOLVE
***AUTHOR   Voorhees, E. A., (LANL)
***DESCRIPTION
```

CNBSL solves the complex band system
 $A * X = B$ or $CTRANS(A) * X = B$
using the factors computed by CNBCO or CNBFA.

On Entry

ABE	COMPLEX(LDA, NC)	the output from CNBCO or CNBFA. NC must be .GE. 2*ML+MU+1 .
LDA	INTEGER	the leading dimension of the array ABE .
N	INTEGER	the order of the original matrix.
ML	INTEGER	number of diagonals below the main diagonal.
MU	INTEGER	number of diagonals above the main diagonal.
IPVT	INTEGER(N)	the pivot vector from CNBCO or CNBFA.
B	COMPLEX(N)	the right hand side vector.
JOB	INTEGER	
	= 0	to solve $A * X = B$.
	= nonzero	to solve $CTRANS(A) * X = B$, where CTRANS(A) is the conjugate transpose.

On Return

B the solution vector X .

Error Condition

A division by zero will occur if the input factor contains a zero on the diagonal. Technically this indicates singularity but it is often caused by improper arguments or improper setting of LDA. It will not occur if the subroutines are called correctly and if CNBCO has set RCOND .GT. 0.0 or CNBFA has set INFO .EQ. 0 .

```

      To compute INVERSE(A) * C where C is a matrix
      with P columns
      CALL CNBCO(ABE,LDA,N,ML,MU,IPVT,RCOND,Z)
      IF (RCOND is too small) GO TO ...
      DO 10 J = 1, P
          CALL CNBSL(ABE,LDA,N,ML,MU,IPVT,C(1,J),0)
      10 CONTINUE

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
              Stewart, LINPACK Users' Guide, SIAM, 1979.
***ROUTINES CALLED CAXPY, CDOTC
***REVISION HISTORY (YYMMDD)
      800730 DATE WRITTEN
      890531 Changed all specific intrinsics to generic. (WRB)
      890831 Modified array declarations. (WRB)
      890831 REVISION DATE from Version 3.2
      891214 Prologue converted to Version 4.0 format. (BAB)
      920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

```

COMBAK

```
SUBROUTINE COMBAK (NM, LOW, IGH, AR, AI, INT, M, ZR, ZI)
***BEGIN PROLOGUE  COMBAK
***PURPOSE  Form the eigenvectors of a complex general matrix from the
             eigenvectors of a upper Hessenberg matrix output from
             COMHES.
***LIBRARY    SLATEC (EISPACK)
***CATEGORY   D4C4
***TYPE       COMPLEX (ELMBAK-S, COMBAK-C)
***KEYWORDS   EIGENVALUES, EIGENVECTORS, EISPACK
***AUTHOR    Smith, B. T., et al.
***DESCRIPTION
```

This subroutine is a translation of the ALGOL procedure COMBAK, NUM. MATH. 12, 349-368(1968) by Martin and Wilkinson. HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 339-358(1971).

This subroutine forms the eigenvectors of a COMPLEX GENERAL matrix by back transforming those of the corresponding upper Hessenberg matrix determined by COMHES.

On INPUT

NM must be set to the row dimension of the two-dimensional array parameters, AR, AI, ZR and ZI, as declared in the calling program dimension statement. NM is an INTEGER variable.

LOW and IGH are two INTEGER variables determined by the balancing subroutine CBAL. If CBAL has not been used, set LOW=1 and IGH equal to the order of the matrix.

AR and AI contain the multipliers which were used in the reduction by COMHES in their lower triangles below the subdiagonal. AR and AI are two-dimensional REAL arrays, dimensioned AR(NM,IGH) and AI(NM,IGH).

INT contains information on the rows and columns interchanged in the reduction by COMHES. Only elements LOW through IGH are used. INT is a one-dimensional INTEGER array, dimensioned INT(IGH).

M is the number of eigenvectors to be back transformed. M is an INTEGER variable.

ZR and ZI contain the real and imaginary parts, respectively, of the eigenvectors to be back transformed in their first M columns. ZR and ZI are two-dimensional REAL arrays, dimensioned ZR(NM,M) and ZI(NM,M).

On OUTPUT

ZR and ZI contain the real and imaginary parts, respectively, of the transformed eigenvectors in their first M columns.

Questions and comments should be directed to B. S. Garbow, APPLIED MATHEMATICS DIVISION, ARGONNE NATIONAL LABORATORY

***REFERENCES B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow,
Y. Ikebe, V. C. Klema and C. B. Moler, Matrix Eigen-
system Routines - EISPACK Guide, Springer-Verlag,
1976.

***ROUTINES CALLED (NONE)

***REVISION HISTORY (YYMMDD)

760101 DATE WRITTEN

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

COMHES

```
SUBROUTINE COMHES (NM, N, LOW, IGH, AR, AI, INT)
***BEGIN PROLOGUE  COMHES
***PURPOSE  Reduce a complex general matrix to complex upper Hessenberg
             form using stabilized elementary similarity
             transformations.
***LIBRARY   SLATEC (EISPACK)
***CATEGORY  D4C1B2
***TYPE      COMPLEX (ELMHES-S, COMHES-C)
***KEYWORDS  EIGENVALUES, EIGENVECTORS, EISPACK
***AUTHOR    Smith, B. T., et al.
***DESCRIPTION
```

This subroutine is a translation of the ALGOL procedure COMHES,
NUM. MATH. 12, 349-368(1968) by Martin and Wilkinson.
HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 339-358(1971).

Given a COMPLEX GENERAL matrix, this subroutine
reduces a submatrix situated in rows and columns
LOW through IGH to upper Hessenberg form by
stabilized elementary similarity transformations.

On INPUT

NM must be set to the row dimension of the two-dimensional
array parameters, AR and AI, as declared in the calling
program dimension statement. NM is an INTEGER variable.

N is the order of the matrix A=(AR,AI). N is an INTEGER
variable. N must be less than or equal to NM.

LOW and IGH are two INTEGER variables determined by the
balancing subroutine CBAL. If CBAL has not been used,
set LOW=1 and IGH equal to the order of the matrix, N.

AR and AI contain the real and imaginary parts, respectively,
of the complex input matrix. AR and AI are two-dimensional
REAL arrays, dimensioned AR(NM,N) and AI(NM,N).

On OUTPUT

AR and AI contain the real and imaginary parts, respectively,
of the upper Hessenberg matrix. The multipliers which
were used in the reduction are stored in the remaining
triangles under the Hessenberg matrix.

INT contains information on the rows and columns
interchanged in the reduction. Only elements LOW through
IGH are used. INT is a one-dimensional INTEGER array,
dimensioned INT(IGH).

Calls CDIV for complex division.

Questions and comments should be directed to B. S. Garbow,
APPLIED MATHEMATICS DIVISION, ARGONNE NATIONAL LABORATORY

***REFERENCES B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow,
Y. Ikebe, V. C. Klema and C. B. Moler, Matrix Eigen-
system Routines - EISPACK Guide, Springer-Verlag,
1976.

***ROUTINES CALLED CDIV

***REVISION HISTORY (YYMMDD)

760101 DATE WRITTEN

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

COMLR

```
SUBROUTINE COMLR (NM, N, LOW, IGH, HR, HI, WR, WI, IERR)
***BEGIN PROLOGUE  COMLR
***PURPOSE  Compute the eigenvalues of a complex upper Hessenberg
            matrix using the modified LR method.
***LIBRARY   SLATEC (EISPACK)
***CATEGORY  D4C2B
***TYPE      COMPLEX (COMLR-C)
***KEYWORDS  EIGENVALUES, EISPACK, LR METHOD
***AUTHOR   Smith, B. T., et al.
***DESCRIPTION
```

This subroutine is a translation of the ALGOL procedure COMLR, NUM. MATH. 12, 369-376(1968) by Martin and Wilkinson. HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 396-403(1971).

This subroutine finds the eigenvalues of a COMPLEX UPPER Hessenberg matrix by the modified LR method.

On INPUT

NM must be set to the row dimension of the two-dimensional array parameters, HR and HI, as declared in the calling program dimension statement. NM is an INTEGER variable.

N is the order of the matrix $H=(HR,HI)$. N is an INTEGER variable. N must be less than or equal to NM.

LOW and IGH are two INTEGER variables determined by the balancing subroutine CBAL. If CBAL has not been used, set LOW=1 and IGH equal to the order of the matrix, N.

HR and HI contain the real and imaginary parts, respectively, of the complex upper Hessenberg matrix. Their lower triangles below the subdiagonal contain the multipliers which were used in the reduction by COMHES, if performed. HR and HI are two-dimensional REAL arrays, dimensioned HR(NM,N) and HI(NM,N).

On OUTPUT

The upper Hessenberg portions of HR and HI have been destroyed. Therefore, they must be saved before calling COMLR if subsequent calculation of eigenvectors is to be performed.

WR and WI contain the real and imaginary parts, respectively, of the eigenvalues of the upper Hessenberg matrix. If an error exit is made, the eigenvalues should be correct for indices IERR+1, IERR+2, ..., N. WR and WI are one-dimensional REAL arrays, dimensioned WR(N) and WI(N).

IERR is an INTEGER flag set to
Zero for normal return,
J if the J-th eigenvalue has not been
 determined after a total of $30*N$ iterations.
The eigenvalues should be correct for indices

IERR+1, IERR+2, ..., N.

Calls CSROOT for complex square root.
Calls CDIV for complex division.

Questions and comments should be directed to B. S. Garbow,
APPLIED MATHEMATICS DIVISION, ARGONNE NATIONAL LABORATORY

***REFERENCES B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow,
Y. Ikebe, V. C. Klema and C. B. Moler, Matrix Eigen-
system Routines - EISPACK Guide, Springer-Verlag,
1976.

***ROUTINES CALLED CDIV, CSROOT

***REVISION HISTORY (YYMMDD)

760101 DATE WRITTEN

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

COMLR2

```
      SUBROUTINE COMLR2 (NM, N, LOW, IGH, INT, HR, HI, WR, WI, ZR, ZI,
+      IERR)
***BEGIN PROLOGUE  COMLR2
***PURPOSE  Compute the eigenvalues and eigenvectors of a complex upper
            Hessenberg matrix using the modified LR method.
***LIBRARY    SLATEC (EISPACK)
***CATEGORY   D4C2B
***TYPE       COMPLEX (COMLR2-C)
***KEYWORDS   EIGENVALUES, EIGENVECTORS, EISPACK, LR METHOD
***AUTHOR    Smith, B. T., et al.
***DESCRIPTION
```

This subroutine is a translation of the ALGOL procedure COMLR2, NUM. MATH. 16, 181-204(1970) by Peters and Wilkinson. HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 372-395(1971).

This subroutine finds the eigenvalues and eigenvectors of a COMPLEX UPPER Hessenberg matrix by the modified LR method. The eigenvectors of a COMPLEX GENERAL matrix can also be found if COMHES has been used to reduce this general matrix to Hessenberg form.

On INPUT

NM must be set to the row dimension of the two-dimensional array parameters, HR, HI, ZR and ZI, as declared in the calling program dimension statement. NM is an INTEGER variable.

N is the order of the matrix H=(HR,HI). N is an INTEGER variable. N must be less than or equal to NM.

LOW and IGH are two INTEGER variables determined by the balancing subroutine CBAL. If CBAL has not been used, set LOW=1 and IGH equal to the order of the matrix, N.

INT contains information on the rows and columns interchanged in the reduction by COMHES, if performed. Only elements LOW through IGH are used. If you want the eigenvectors of a complex general matrix, leave INT as it came from COMHES. If the eigenvectors of the Hessenberg matrix are desired, set INT(J)=J for these elements. INT is a one-dimensional INTEGER array, dimensioned INT(IGH).

HR and HI contain the real and imaginary parts, respectively, of the complex upper Hessenberg matrix. Their lower triangles below the subdiagonal contain the multipliers which were used in the reduction by COMHES, if performed. If the eigenvectors of a complex general matrix are desired, leave these multipliers in the lower triangles. If the eigenvectors of the Hessenberg matrix are desired, these elements must be set to zero. HR and HI are two-dimensional REAL arrays, dimensioned HR(NM,N) and HI(NM,N).

On OUTPUT

The upper Hessenberg portions of HR and HI have been destroyed, but the location HR(1,1) contains the norm of the triangularized matrix.

WR and WI contain the real and imaginary parts, respectively, of the eigenvalues of the upper Hessenberg matrix. If an error exit is made, the eigenvalues should be correct for indices IERR+1, IERR+2, ..., N. WR and WI are one-dimensional REAL arrays, dimensioned WR(N) and WI(N).

ZR and ZI contain the real and imaginary parts, respectively, of the eigenvectors. The eigenvectors are unnormalized. If an error exit is made, none of the eigenvectors has been found. ZR and ZI are two-dimensional REAL arrays, dimensioned ZR(NM,N) and ZI(NM,N).

IERR is an INTEGER flag set to
Zero for normal return,
J if the J-th eigenvalue has not been determined after a total of 30*N iterations. The eigenvalues should be correct for indices IERR+1, IERR+2, ..., N, but no eigenvectors are computed.

Calls CSROOT for complex square root.
Calls CDIV for complex division.

Questions and comments should be directed to B. S. Garbow,
APPLIED MATHEMATICS DIVISION, ARGONNE NATIONAL LABORATORY

***REFERENCES B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow,
Y. Ikebe, V. C. Klema and C. B. Moler, Matrix Eigen-
system Routines - EISPACK Guide, Springer-Verlag,
1976.

***ROUTINES CALLED CDIV, CSROOT

***REVISION HISTORY (YYMMDD)

760101 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

COMQR

```
SUBROUTINE COMQR (NM, N, LOW, IGH, HR, HI, WR, WI, IERR)
***BEGIN PROLOGUE  COMQR
***PURPOSE  Compute the eigenvalues of complex upper Hessenberg matrix
             using the QR method.
***LIBRARY   SLATEC (EISPACK)
***CATEGORY  D4C2B
***TYPE      COMPLEX (HQR-S, COMQR-C)
***KEYWORDS  EIGENVALUES, EIGENVECTORS, EISPACK
***AUTHOR    Smith, B. T., et al.
***DESCRIPTION
```

This subroutine is a translation of a unitary analogue of the ALGOL procedure COMLR, NUM. MATH. 12, 369-376(1968) by Martin and Wilkinson.
HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 396-403(1971).
The unitary analogue substitutes the QR algorithm of Francis (COMP. JOUR. 4, 332-345(1962)) for the LR algorithm.

This subroutine finds the eigenvalues of a COMPLEX upper Hessenberg matrix by the QR method.

On INPUT

NM must be set to the row dimension of the two-dimensional array parameters, HR and HI, as declared in the calling program dimension statement. NM is an INTEGER variable.

N is the order of the matrix H=(HR,HI). N is an INTEGER variable. N must be less than or equal to NM.

LOW and IGH are two INTEGER variables determined by the balancing subroutine CBAL. If CBAL has not been used, set LOW=1 and IGH equal to the order of the matrix, N.

HR and HI contain the real and imaginary parts, respectively, of the complex upper Hessenberg matrix. Their lower triangles below the subdiagonal contain information about the unitary transformations used in the reduction by CORTH, if performed. HR and HI are two-dimensional REAL arrays, dimensioned HR(NM,N) and HI(NM,N).

On OUTPUT

The upper Hessenberg portions of HR and HI have been destroyed. Therefore, they must be saved before calling COMQR if subsequent calculation of eigenvectors is to be performed.

WR and WI contain the real and imaginary parts, respectively, of the eigenvalues of the upper Hessenberg matrix. If an error exit is made, the eigenvalues should be correct for indices IERR+1, IERR+2, ..., N. WR and WI are one-dimensional REAL arrays, dimensioned WR(N) and WI(N).

IERR is an INTEGER flag set to
Zero for normal return,

J if the J-th eigenvalue has not been
 determined after a total of 30*N iterations.
 The eigenvalues should be correct for indices
 IERR+1, IERR+2, ..., N.

Calls CSROOT for complex square root.
Calls PYTHAG(A,B) for $\text{sqrt}(A^2 + B^2)$.
Calls CDIV for complex division.

Questions and comments should be directed to B. S. Garbow,
APPLIED MATHEMATICS DIVISION, ARGONNE NATIONAL LABORATORY

***REFERENCES B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow,
 Y. Ikebe, V. C. Klema and C. B. Moler, Matrix Eigen-
 system Routines - EISPACK Guide, Springer-Verlag,
 1976.

***ROUTINES CALLED CDIV, CSROOT, PYTHAG

***REVISION HISTORY (YMMDD)

760101 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

COMQR2

```
      SUBROUTINE COMQR2 (NM, N, LOW, IGH, ORTR, ORTI, HR, HI, WR, WI,  
+      ZR, ZI, IERR)  
***BEGIN PROLOGUE  COMQR2  
***PURPOSE  Compute the eigenvalues and eigenvectors of a complex upper  
            Hessenberg matrix.  
***LIBRARY   SLATEC (EISPACK)  
***CATEGORY  D4C2B  
***TYPE      COMPLEX (HQR2-S, COMQR2-C)  
***KEYWORDS  EIGENVALUES, EIGENVECTORS, EISPACK  
***AUTHOR   Smith, B. T., et al.  
***DESCRIPTION
```

This subroutine is a translation of a unitary analogue of the ALGOL procedure COMLR2, NUM. MATH. 16, 181-204(1970) by Peters and Wilkinson.
HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 372-395(1971).
The unitary analogue substitutes the QR algorithm of Francis (COMP. JOUR. 4, 332-345(1962)) for the LR algorithm.

This subroutine finds the eigenvalues and eigenvectors of a COMPLEX UPPER Hessenberg matrix by the QR method. The eigenvectors of a COMPLEX GENERAL matrix can also be found if CORTH has been used to reduce this general matrix to Hessenberg form.

On INPUT

NM must be set to the row dimension of the two-dimensional array parameters, HR, HI, ZR, and ZI, as declared in the calling program dimension statement. NM is an INTEGER variable.

N is the order of the matrix H=(HR,HI). N is an INTEGER variable. N must be less than or equal to NM.

LOW and IGH are two INTEGER variables determined by the balancing subroutine CBAL. If CBAL has not been used, set LOW=1 and IGH equal to the order of the matrix, N.

ORTR and ORTI contain information about the unitary transformations used in the reduction by CORTH, if performed. Only elements LOW through IGH are used. If the eigenvectors of the Hessenberg matrix are desired, set ORTR(J) and ORTI(J) to 0.0E0 for these elements. ORTR and ORTI are one-dimensional REAL arrays, dimensioned ORTR(IGH) and ORTI(IGH).

HR and HI contain the real and imaginary parts, respectively, of the complex upper Hessenberg matrix. Their lower triangles below the subdiagonal contain information about the unitary transformations used in the reduction by CORTH, if performed. If the eigenvectors of the Hessenberg matrix are desired, these elements may be arbitrary. HR and HI are two-dimensional REAL arrays, dimensioned HR(NM,N) and HI(NM,N).

On OUTPUT

ORTR, ORTI, and the upper Hessenberg portions of HR and HI have been destroyed.

WR and WI contain the real and imaginary parts, respectively, of the eigenvalues of the upper Hessenberg matrix. If an error exit is made, the eigenvalues should be correct for indices IERR+1, IERR+2, ..., N. WR and WI are one-dimensional REAL arrays, dimensioned WR(N) and WI(N).

ZR and ZI contain the real and imaginary parts, respectively, of the eigenvectors. The eigenvectors are unnormalized. If an error exit is made, none of the eigenvectors has been found. ZR and ZI are two-dimensional REAL arrays, dimensioned ZR(NM,N) and ZI(NM,N).

IERR is an INTEGER flag set to
Zero for normal return,
J if the J-th eigenvalue has not been
determined after a total of 30*N iterations.
The eigenvalues should be correct for indices
IERR+1, IERR+2, ..., N, but no eigenvectors are
computed.

Calls CSROOT for complex square root.
Calls PYTHAG(A,B) for $\sqrt{A^2 + B^2}$.
Calls CDIV for complex division.

Questions and comments should be directed to B. S. Garbow,
APPLIED MATHEMATICS DIVISION, ARGONNE NATIONAL LABORATORY

***REFERENCES B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow,
Y. Ikebe, V. C. Klema and C. B. Moler, Matrix Eigen-
system Routines - EISPACK Guide, Springer-Verlag,
1976.

***ROUTINES CALLED CDIV, CSROOT, PYTHAG

***REVISION HISTORY (YYMMDD)

760101 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CORTB

```
      SUBROUTINE CORTB (NM, LOW, IGH, AR, AI, ORTR, ORTI, M, ZR, ZI)
***BEGIN PROLOGUE  CORTB
***PURPOSE  Form the eigenvectors of a complex general matrix from
            eigenvectors of upper Hessenberg matrix output from
            CORTH.
***LIBRARY    SLATEC (EISPACK)
***CATEGORY  D4C4
***TYPE       COMPLEX (ORTBAK-S, CORTB-C)
***KEYWORDS  EIGENVALUES, EIGENVECTORS, EISPACK
***AUTHOR  Smith, B. T., et al.
***DESCRIPTION
```

This subroutine is a translation of a complex analogue of the ALGOL procedure ORTBAK, NUM. MATH. 12, 349-368(1968) by Martin and Wilkinson. HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 339-358(1971).

This subroutine forms the eigenvectors of a COMPLEX GENERAL matrix by back transforming those of the corresponding upper Hessenberg matrix determined by CORTH.

On INPUT

NM must be set to the row dimension of the two-dimensional array parameters, AR, AI, ZR, and ZI, as declared in the calling program dimension statement. NM is an INTEGER variable.

LOW and IGH are two INTEGER variables determined by the balancing subroutine CBAL. If CBAL has not been used, set LOW=1 and IGH equal to the order of the matrix.

AR and AI contain information about the unitary transformations used in the reduction by CORTH in their strict lower triangles. AR and AI are two-dimensional REAL arrays, dimensioned AR(NM,IGH) and AI(NM,IGH).

ORTR and ORTI contain further information about the unitary transformations used in the reduction by CORTH. Only elements LOW through IGH are used. ORTR and ORTI are one-dimensional REAL arrays, dimensioned ORTR(IGH) and ORTI(IGH).

M is the number of columns of Z=(ZR,ZI) to be back transformed. M is an INTEGER variable.

ZR and ZI contain the real and imaginary parts, respectively, of the eigenvectors to be back transformed in their first M columns. ZR and ZI are two-dimensional REAL arrays, dimensioned ZR(NM,M) and ZI(NM,M).

On OUTPUT

ZR and ZI contain the real and imaginary parts, respectively, of the transformed eigenvectors in their first M columns.

ORTR and ORTI have been altered.

Note that CORTB preserves vector Euclidean norms.

Questions and comments should be directed to B. S. Garbow,
APPLIED MATHEMATICS DIVISION, ARGONNE NATIONAL LABORATORY

***REFERENCES B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow,
Y. Ikebe, V. C. Klema and C. B. Moler, Matrix Eigen-
system Routines - EISPACK Guide, Springer-Verlag,
1976.

***ROUTINES CALLED (NONE)

***REVISION HISTORY (YYMMDD)

760101 DATE WRITTEN

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CORTH

```
SUBROUTINE CORTH (NM, N, LOW, IGH, AR, AI, ORTR, ORTI)
***BEGIN PROLOGUE  CORTH
***PURPOSE  Reduce a complex general matrix to complex upper Hessenberg
             form using unitary similarity transformations.
***LIBRARY   SLATEC (EISPACK)
***CATEGORY  D4C1B2
***TYPE      COMPLEX (ORTHES-S, CORTH-C)
***KEYWORDS  EIGENVALUES, EIGENVECTORS, EISPACK
***AUTHOR   Smith, B. T., et al.
***DESCRIPTION
```

This subroutine is a translation of a complex analogue of the ALGOL procedure ORTHES, NUM. MATH. 12, 349-368(1968) by Martin and Wilkinson.
HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 339-358(1971).

Given a COMPLEX GENERAL matrix, this subroutine reduces a submatrix situated in rows and columns LOW through IGH to upper Hessenberg form by unitary similarity transformations.

On INPUT

NM must be set to the row dimension of the two-dimensional array parameters, AR and AI, as declared in the calling program dimension statement. NM is an INTEGER variable.

N is the order of the matrix A=(AR,AI). N is an INTEGER variable. N must be less than or equal to NM.

LOW and IGH are two INTEGER variables determined by the balancing subroutine CBAL. If CBAL has not been used, set LOW=1 and IGH equal to the order of the matrix, N.

AR and AI contain the real and imaginary parts, respectively, of the complex input matrix. AR and AI are two-dimensional REAL arrays, dimensioned AR(NM,N) and AI(NM,N).

On OUTPUT

AR and AI contain the real and imaginary parts, respectively, of the Hessenberg matrix. Information about the unitary transformations used in the reduction is stored in the remaining triangles under the Hessenberg matrix.

ORTR and ORTI contain further information about the unitary transformations. Only elements LOW through IGH are used. ORTR and ORTI are one-dimensional REAL arrays, dimensioned ORTR(IGH) and ORTI(IGH).

Calls PYTHAG(A,B) for $\sqrt{A^2 + B^2}$.

Questions and comments should be directed to B. S. Garbow,
APPLIED MATHEMATICS DIVISION, ARGONNE NATIONAL LABORATORY

```
***REFERENCES  B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow,  
                Y. Ikebe, V. C. Klema and C. B. Moler, Matrix Eigen-  
                system Routines - EISPACK Guide, Springer-Verlag,  
                1976.  
***ROUTINES CALLED  PYTHAG  
***REVISION HISTORY  (YYMMDD)  
    760101  DATE WRITTEN  
    890831  Modified array declarations.  (WRB)  
    890831  REVISION DATE from Version 3.2  
    891214  Prologue converted to Version 4.0 format.  (BAB)  
    920501  Reformatted the REFERENCES section.  (WRB)  
END PROLOGUE
```

COSDG

```
      FUNCTION COSDG (X)
***BEGIN PROLOGUE  COSDG
***PURPOSE  Compute the cosine of an argument in degrees.
***LIBRARY  SLATEC (FNLIB)
***CATEGORY  C4A
***TYPE      SINGLE PRECISION (COSDG-S, DCOSDG-D)
***KEYWORDS  COSINE, DEGREES, ELEMENTARY FUNCTIONS, FNLIB,
              TRIGONOMETRIC
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION

      COSDG(X) evaluates the cosine for real X in degrees.

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      770601  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890531  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      END PROLOGUE
```

COSQB

```
      SUBROUTINE COSQB (N, X, WSAVE)
***BEGIN PROLOGUE  COSQB
***PURPOSE  Compute the unnormalized inverse cosine transform.
***LIBRARY   SLATEC (FFTPACK)
***CATEGORY  J1A3
***TYPE      SINGLE PRECISION (COSQB-S)
***KEYWORDS  FFTPACK, INVERSE COSINE FOURIER TRANSFORM
***AUTHOR   Swarztrauber, P. N., (NCAR)
***DESCRIPTION
```

Subroutine COSQB computes the fast Fourier transform of quarter wave data. That is, COSQB computes a sequence from its representation in terms of a cosine series with odd wave numbers. The transform is defined below at output parameter X.

COSQB is the unnormalized inverse of COSQF since a call of COSQB followed by a call of COSQF will multiply the input sequence X by $4*N$.

The array WSAVE which is used by subroutine COSQB must be initialized by calling subroutine COSQI(N,WSAVE).

Input Parameters

N the length of the array X to be transformed. The method is most efficient when N is a product of small primes.

X an array which contains the sequence to be transformed

WSAVE a work array which must be dimensioned at least $3*N+15$ in the program that calls COSQB. The WSAVE array must be initialized by calling subroutine COSQI(N,WSAVE), and a different WSAVE array must be used for each different value of N. This initialization does not have to be repeated so long as N remains unchanged. Thus subsequent transforms can be obtained faster than the first.

Output Parameters

X For $I=1, \dots, N$

$X(I) =$ the sum from $K=1$ to $K=N$ of

$$2*X(K)*\cos((2*K-1)*(I-1)*\pi/(2*N))$$

A call of COSQB followed by a call of COSQF will multiply the sequence X by $4*N$. Therefore COSQF is the unnormalized inverse of COSQB.

WSAVE contains initialization calculations which must not be destroyed between calls of COSQB or COSQF.

```
***REFERENCES  P. N. Swarztrauber, Vectorizing the FFTs, in Parallel
                Computations (G. Rodrigue, ed.), Academic Press,
```

1982, pp. 51-83.

```
***ROUTINES CALLED  COSQB1
***REVISION HISTORY  (YYMMDD)
 790601  DATE WRITTEN
 830401  Modified to use SLATEC library source file format.
 860115  Modified by Ron Boisvert to adhere to Fortran 77 by
        (a) changing dummy array size declarations (1) to (*),
        (b) changing definition of variable TSQRT2 by using
            FORTRAN intrinsic function SQRT instead of a DATA
            statement.
 861211  REVISION DATE from Version 3.2
 881128  Modified by Dick Valent to meet prologue standards.
 891214  Prologue converted to Version 4.0 format.  (BAB)
 920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

COSQF

```
      SUBROUTINE COSQF (N, X, WSAVE)
***BEGIN PROLOGUE  COSQF
***PURPOSE  Compute the forward cosine transform with odd wave numbers.
***LIBRARY   SLATEC (FFTPACK)
***CATEGORY  J1A3
***TYPE      SINGLE PRECISION (COSQF-S)
***KEYWORDS  COSINE FOURIER TRANSFORM, FFTPACK
***AUTHOR   Swarztrauber, P. N., (NCAR)
***DESCRIPTION
```

Subroutine COSQF computes the fast Fourier transform of quarter wave data. That is, COSQF computes the coefficients in a cosine series representation with only odd wave numbers. The transform is defined below at Output Parameter X

COSQF is the unnormalized inverse of COSQB since a call of COSQF followed by a call of COSQB will multiply the input sequence X by $4*N$.

The array WSAVE which is used by subroutine COSQF must be initialized by calling subroutine COSQI(N,WSAVE).

Input Parameters

N the length of the array X to be transformed. The method is most efficient when N is a product of small primes.

X an array which contains the sequence to be transformed

WSAVE a work array which must be dimensioned at least $3*N+15$ in the program that calls COSQF. The WSAVE array must be initialized by calling subroutine COSQI(N,WSAVE), and a different WSAVE array must be used for each different value of N. This initialization does not have to be repeated so long as N remains unchanged. Thus subsequent transforms can be obtained faster than the first.

Output Parameters

X For $I=1, \dots, N$

$$X(I) = X(1) \text{ plus the sum from } K=2 \text{ to } K=N \text{ of}$$
$$2*X(K)*\cos((2*I-1)*(K-1)*\pi/(2*N))$$

A call of COSQF followed by a call of COSQB will multiply the sequence X by $4*N$. Therefore COSQB is the unnormalized inverse of COSQF.

WSAVE contains initialization calculations which must not be destroyed between calls of COSQF or COSQB.

```
***REFERENCES  P. N. Swarztrauber, Vectorizing the FFTs, in Parallel
                Computations (G. Rodrigue, ed.), Academic Press,
```

1982, pp. 51-83.

```
***ROUTINES CALLED  COSQF1
***REVISION HISTORY  (YYMMDD)
 790601  DATE WRITTEN
 830401  Modified to use SLATEC library source file format.
 860115  Modified by Ron Boisvert to adhere to Fortran 77 by
        (a) changing dummy array size declarations (1) to (*),
        (b) changing definition of variable SQRT2 by using
            FORTRAN intrinsic function SQRT instead of a DATA
            statement.
 861211  REVISION DATE from Version 3.2
 881128  Modified by Dick Valent to meet prologue standards.
 891214  Prologue converted to Version 4.0 format.  (BAB)
 920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

COSQI

```
      SUBROUTINE COSQI (N, WSAVE)
***BEGIN PROLOGUE  COSQI
***PURPOSE  Initialize a work array for COSQF and COSQB.
***LIBRARY  SLATEC (FFTPACK)
***CATEGORY  J1A3
***TYPE      SINGLE PRECISION (COSQI-S)
***KEYWORDS  COSINE FOURIER TRANSFORM, FFTPACK
***AUTHOR  Swarztrauber, P. N., (NCAR)
***DESCRIPTION
```

Subroutine COSQI initializes the work array WSAVE which is used in both COSQF1 and COSQB1. The prime factorization of N together with a tabulation of the trigonometric functions are computed and stored in WSAVE.

Input Parameter

N the length of the array to be transformed. The method is most efficient when N is a product of small primes.

Output Parameter

WSAVE a work array which must be dimensioned at least 3*N+15. The same work array can be used for both COSQF1 and COSQB1 as long as N remains unchanged. Different WSAVE arrays are required for different values of N. The contents of WSAVE must not be changed between calls of COSQF1 or COSQB1.

```
***REFERENCES  P. N. Swarztrauber, Vectorizing the FFTs, in Parallel
                Computations (G. Rodrigue, ed.), Academic Press,
                1982, pp. 51-83.
***ROUTINES CALLED  RFFTI
***REVISION HISTORY  (YYMMDD)
   790601  DATE WRITTEN
   830401  Modified to use SLATEC library source file format.
   860115  Modified by Ron Boisvert to adhere to Fortran 77 by
           (a) changing dummy array size declarations (1) to (*),
           (b) changing references to intrinsic function FLOAT
               to REAL, and
           (c) changing definition of variable PIH by using
               FORTRAN intrinsic function ATAN instead of a DATA
               statement.
   881128  Modified by Dick Valent to meet prologue standards.
   890531  Changed all specific intrinsics to generic.  (WRB)
   890531  REVISION DATE from Version 3.2
   891214  Prologue converted to Version 4.0 format.  (BAB)
   920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

COST

```
      SUBROUTINE COST (N, X, WSAVE)
***BEGIN PROLOGUE  COST
***PURPOSE  Compute the cosine transform of a real, even sequence.
***LIBRARY   SLATEC (FFTPACK)
***CATEGORY  J1A3
***TYPE      SINGLE PRECISION (COST-S)
***KEYWORDS  COSINE FOURIER TRANSFORM, FFTPACK
***AUTHOR   Swarztrauber, P. N., (NCAR)
***DESCRIPTION
```

Subroutine COST computes the discrete Fourier cosine transform of an even sequence X(I). The transform is defined below at output parameter X.

COST is the unnormalized inverse of itself since a call of COST followed by another call of COST will multiply the input sequence X by $2*(N-1)$. The transform is defined below at output parameter X.

The array WSAVE which is used by subroutine COST must be initialized by calling subroutine COSTI(N,WSAVE).

Input Parameters

N the length of the sequence X. N must be greater than 1. The method is most efficient when N-1 is a product of small primes.

X an array which contains the sequence to be transformed

WSAVE a work array which must be dimensioned at least $3*N+15$ in the program that calls COST. The WSAVE array must be initialized by calling subroutine COSTI(N,WSAVE), and a different WSAVE array must be used for each different value of N. This initialization does not have to be repeated so long as N remains unchanged. Thus subsequent transforms can be obtained faster than the first.

Output Parameters

X For $I=1, \dots, N$

$$X(I) = X(1) + (-1)^{(I-1)} * X(N) \\ + \text{the sum from } K=2 \text{ to } K=N-1 \\ 2 * X(K) * \cos((K-1) * (I-1) * \pi / (N-1))$$

A call of COST followed by another call of COST will multiply the sequence X by $2*(N-1)$. Hence COST is the unnormalized inverse of itself.

WSAVE contains initialization calculations which must not be destroyed between calls of COST.

```
***REFERENCES  P. N. Swarztrauber, Vectorizing the FFTs, in Parallel
```

Computations (G. Rodrigue, ed.), Academic Press,
1982, pp. 51-83.

***ROUTINES CALLED RFFTF

***REVISION HISTORY (YYMMDD)

790601 DATE WRITTEN

830401 Modified to use SLATEC library source file format.

860115 Modified by Ron Boisvert to adhere to Fortran 77 by
changing dummy array size declarations (1) to (*)

861211 REVISION DATE from Version 3.2

881128 Modified by Dick Valent to meet prologue standards.

891214 Prologue converted to Version 4.0 format. (BAB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

COSTI

```
      SUBROUTINE COSTI (N, WSAVE)
***BEGIN PROLOGUE  COSTI
***PURPOSE  Initialize a work array for COST.
***LIBRARY   SLATEC (FFTPACK)
***CATEGORY  J1A3
***TYPE      SINGLE PRECISION (COSTI-S)
***KEYWORDS  COSINE FOURIER TRANSFORM, FFTPACK
***AUTHOR   Swarztrauber, P. N., (NCAR)
***DESCRIPTION
```

Subroutine COSTI initializes the array WSAVE which is used in subroutine COST. The prime factorization of N together with a tabulation of the trigonometric functions are computed and stored in WSAVE.

Input Parameter

N the length of the sequence to be transformed. The method is most efficient when N-1 is a product of small primes.

Output Parameter

WSAVE a work array which must be dimensioned at least 3*N+15. Different WSAVE arrays are required for different values of N. The contents of WSAVE must not be changed between calls of COST.

```
***REFERENCES  P. N. Swarztrauber, Vectorizing the FFTs, in Parallel
                Computations (G. Rodrigue, ed.), Academic Press,
                1982, pp. 51-83.
***ROUTINES CALLED  RFFFTI
***REVISION HISTORY  (YYMMDD)
   790601  DATE WRITTEN
   830401  Modified to use SLATEC library source file format.
   860115  Modified by Ron Boisvert to adhere to Fortran 77 by
          (a) changing dummy array size declarations (1) to (*),
          (b) changing references to intrinsic function FLOAT
              to REAL, and
          (c) changing definition of variable PI by using
              FORTRAN intrinsic function ATAN instead of a DATA
              statement.
   881128  Modified by Dick Valent to meet prologue standards.
   890531  Changed all specific intrinsics to generic.  (WRB)
   890531  REVISION DATE from Version 3.2
   891214  Prologue converted to Version 4.0 format.  (BAB)
   920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

COT

```
FUNCTION COT (X)
***BEGIN PROLOGUE  COT
***PURPOSE  Compute the cotangent.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C4A
***TYPE      SINGLE PRECISION (COT-S, DCOT-D, CCOT-C)
***KEYWORDS  COTANGENT, ELEMENTARY FUNCTIONS, FNLIB, TRIGONOMETRIC
***AUTHOR    Fullerton, W., (LANL)
***DESCRIPTION
```

COT(X) calculates the cotangent of the real argument X. X is in units of radians.

Series for COT	on the interval	0.	to	6.25000D-02
			with weighted error	3.76E-17
			log weighted error	16.42
			significant figures required	15.51
			decimal places required	16.88

```
***REFERENCES  (NONE)
***ROUTINES CALLED  CSEVL, INITS, R1MACH, XERMSG
***REVISION HISTORY  (YYMMDD)
  770601  DATE WRITTEN
  890531  Changed all specific intrinsics to generic.  (WRB)
  890531  REVISION DATE from Version 3.2
  891214  Prologue converted to Version 4.0 format.  (BAB)
  900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
  920618  Removed space from variable names.  (RWC, WRB)
END PROLOGUE
```

CPBCO

```
SUBROUTINE CPBCO (ABD, LDA, N, M, RCOND, Z, INFO)
***BEGIN PROLOGUE  CPBCO
***PURPOSE  Factor a complex Hermitian positive definite matrix stored
             in band form and estimate the condition number of the
             matrix.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2D2
***TYPE      COMPLEX (SPBCO-S, DPBCO-D, CPBCO-C)
***KEYWORDS  BANDED, CONDITION NUMBER, LINEAR ALGEBRA, LINPACK,
             MATRIX FACTORIZATION, POSITIVE DEFINITE
***AUTHOR  Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CPBCO factors a complex Hermitian positive definite matrix stored in band form and estimates the condition of the matrix.

If RCOND is not needed, CPBFA is slightly faster.
To solve $A \cdot X = B$, follow CPBCO by CPBSL.
To compute $\text{INVERSE}(A) \cdot C$, follow CPBCO by CPBSL.
To compute $\text{DETERMINANT}(A)$, follow CPBCO by CPBDI.

On Entry

ABD COMPLEX(LDA, N)
 the matrix to be factored. The columns of the upper triangle are stored in the columns of ABD and the diagonals of the upper triangle are stored in the rows of ABD. See the comments below for details.

LDA INTEGER
 the leading dimension of the array ABD.
 LDA must be $\geq M + 1$.

N INTEGER
 the order of the matrix A.

M INTEGER
 the number of diagonals above the main diagonal.
 0 $\leq M < N$.

On Return

ABD an upper triangular matrix R, stored in band form, so that $A = \text{CTRANS}(R) \cdot R$.
 If INFO $\neq 0$, the factorization is not complete.

RCOND REAL
 an estimate of the reciprocal condition of A.
 For the system $A \cdot X = B$, relative perturbations in A and B of size EPSILON may cause relative perturbations in X of size $\text{EPSILON}/\text{RCOND}$.
 If RCOND is so small that the logical expression
 $1.0 + \text{RCOND} \text{ .EQ. } 1.0$
 is true, then A may be singular to working precision. In particular, RCOND is zero if exact singularity is detected or the estimate

underflows. If INFO .NE. 0 , RCOND is unchanged.

Z COMPLEX(N)
a work vector whose contents are usually unimportant.
If A is singular to working precision, then Z is
an approximate null vector in the sense that
 $\text{NORM}(A*Z) = \text{RCOND}*\text{NORM}(A)*\text{NORM}(Z)$.
If INFO .NE. 0 , Z is unchanged.

INFO INTEGER
= 0 for normal return.
= K signals an error condition. The leading minor
 of order K is not positive definite.

Band Storage

If A is a Hermitian positive definite band matrix,
the following program segment will set up the input.

```
      M = (band width above diagonal)
      DO 20 J = 1, N
        I1 = MAX(1, J-M)
        DO 10 I = I1, J
          K = I-J+M+1
          ABD(K,J) = A(I,J)
        10 CONTINUE
      20 CONTINUE
```

This uses M + 1 rows of A , except for the M by M
upper left triangle, which is ignored.

Example: If the original matrix is

```
11 12 13  0  0  0
12 22 23 24  0  0
13 23 33 34 35  0
 0 24 34 44 45 46
 0  0 35 45 55 56
 0  0  0 46 56 66
```

then N = 6 , M = 2 and ABD should contain

```
  *  * 13 24 35 46
  * 12 23 34 45 56
11 22 33 44 55 66
```

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CAXPY, CDOTC, CPBFA, CSSCAL, SCASUM

***REVISION HISTORY (YYMMDD)

780814 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900326 Removed duplicate information from DESCRIPTION section.
(WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CPBDI

```
      SUBROUTINE CPBDI (ABD, LDA, N, M, DET)
***BEGIN PROLOGUE  CPBDI
***PURPOSE  Compute the determinant of a complex Hermitian positive
             definite band matrix using the factors computed by CPBCO or
             CPBFA.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D3D2
***TYPE      COMPLEX (SPBDI-S, DPBDI-D, CPBDI-C)
***KEYWORDS  BANDED, DETERMINANT, INVERSE, LINEAR ALGEBRA, LINPACK,
             MATRIX, POSITIVE DEFINITE
***AUTHOR   Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CPBDI computes the determinant
of a complex Hermitian positive definite band matrix
using the factors computed by CPBCO or CPBFA.
If the inverse is needed, use CPBSL N times.

On Entry

ABD COMPLEX(LDA, N)
 the output from CPBCO or CPBFA.

LDA INTEGER
 the leading dimension of the array ABD .

N INTEGER
 the order of the matrix A .

M INTEGER
 the number of diagonals above the main diagonal.

On Return

DET REAL(2)
 determinant of original matrix in the form
 determinant = DET(1) * 10.0**DET(2)
 with 1.0 .LE. DET(1) .LT. 10.0
 or DET(1) .EQ. 0.0 .

```
***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
                Stewart, LINPACK Users' Guide, SIAM, 1979.
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
  780814  DATE WRITTEN
  890831  Modified array declarations.  (WRB)
  890831  REVISION DATE from Version 3.2
  891214  Prologue converted to Version 4.0 format.  (BAB)
  900326  Removed duplicate information from DESCRIPTION section.
          (WRB)
  920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

CPBFA

```
SUBROUTINE CPBFA (ABD, LDA, N, M, INFO)
***BEGIN PROLOGUE  CPBFA
***PURPOSE  Factor a complex Hermitian positive definite matrix stored
             in band form.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2D2
***TYPE      COMPLEX (SPBFA-S, DPBFA-D, CPBFA-C)
***KEYWORDS  BANDED, LINEAR ALGEBRA, LINPACK, MATRIX FACTORIZATION,
             POSITIVE DEFINITE
***AUTHOR    Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CPBFA factors a complex Hermitian positive definite matrix stored in band form.

CPBFA is usually called by CPBCO, but it can be called directly with a saving in time if RCOND is not needed.

On Entry

ABD COMPLEX(LDA, N)
 the matrix to be factored. The columns of the upper triangle are stored in the columns of ABD and the diagonals of the upper triangle are stored in the rows of ABD . See the comments below for details.

LDA INTEGER
 the leading dimension of the array ABD .
LDA must be .GE. M + 1 .

N INTEGER
 the order of the matrix A .

M INTEGER
 the number of diagonals above the main diagonal.
0 .LE. M .LT. N .

On Return

ABD an upper triangular matrix R , stored in band form, so that $A = CTRANS(R)*R$.

INFO INTEGER
 = 0 for normal return.
 = K if the leading minor of order K is not positive definite.

Band Storage

If A is a Hermitian positive definite band matrix, the following program segment will set up the input.

```
M = (band width above diagonal)
DO 20 J = 1, N
  I1 = MAX(1, J-M)
  DO 10 I = I1, J
```

```

                K = I-J+M+1
                ABD(K,J) = A(I,J)
10      CONTINUE
20 CONTINUE

```

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CDOTC

***REVISION HISTORY (YYMMDD)

```

780814  DATE WRITTEN
890531  Changed all specific intrinsics to generic.  (WRB)
890831  Modified array declarations.  (WRB)
890831  REVISION DATE from Version 3.2
891214  Prologue converted to Version 4.0 format.  (BAB)
900326  Removed duplicate information from DESCRIPTION section.
        (WRB)
920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE

```

CPBSL

```
SUBROUTINE CPBSL (ABD, LDA, N, M, B)
***BEGIN PROLOGUE  CPBSL
***PURPOSE  Solve the complex Hermitian positive definite band system
              using the factors computed by CPBCO or CPBFA.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2D2
***TYPE      COMPLEX (SPBSL-S, DPBSL-D, CPBSL-C)
***KEYWORDS  BANDED, LINEAR ALGEBRA, LINPACK, MATRIX,
              POSITIVE DEFINITE, SOLVE
***AUTHOR   Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CPBSL solves the complex Hermitian positive definite band system $A \cdot X = B$ using the factors computed by CPBCO or CPBFA.

On Entry

ABD COMPLEX(LDA, N)
 the output from CPBCO or CPBFA.

LDA INTEGER
 the leading dimension of the array ABD .

N INTEGER
 the order of the matrix A .

M INTEGER
 the number of diagonals above the main diagonal.

B COMPLEX(N)
 the right hand side vector.

On Return

B the solution vector X .

Error Condition

A division by zero will occur if the input factor contains a zero on the diagonal. Technically this indicates singularity but it is usually caused by improper subroutine arguments. It will not occur if the subroutines are called correctly and INFO .EQ. 0 .

To compute $\text{INVERSE}(A) \cdot C$ where C is a matrix with P columns

```
CALL CPBCO(ABD,LDA,N,RCOND,Z,INFO)
IF (RCOND is too small .OR. INFO .NE. 0) GO TO ...
DO 10 J = 1, P
    CALL CPBSL(ABD,LDA,N,C(1,J))
10 CONTINUE
```

```
***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
                Stewart, LINPACK Users' Guide, SIAM, 1979.
***ROUTINES CALLED  CAXPY, CDOTC
```

***REVISION HISTORY (YYMMDD)
780814 DATE WRITTEN
890531 Changed all specific intrinsics to generic. (WRB)
890831 Modified array declarations. (WRB)
890831 REVISION DATE from Version 3.2
891214 Prologue converted to Version 4.0 format. (BAB)
900326 Removed duplicate information from DESCRIPTION section.
(WRB)
920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

CPOCO

```
SUBROUTINE CPOCO (A, LDA, N, RCOND, Z, INFO)
***BEGIN PROLOGUE  CPOCO
***PURPOSE  Factor a complex Hermitian positive definite matrix
             and estimate the condition number of the matrix.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2D1B
***TYPE      COMPLEX (SPOCO-S, DPOCO-D, CPOCO-C)
***KEYWORDS  CONDITION NUMBER, LINEAR ALGEBRA, LINPACK,
             MATRIX FACTORIZATION, POSITIVE DEFINITE
***AUTHOR    Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CPOCO factors a complex Hermitian positive definite matrix and estimates the condition of the matrix.

If RCOND is not needed, CPOFA is slightly faster.
To solve $A \cdot X = B$, follow CPOCO by CPOSL.
To compute $\text{INVERSE}(A) \cdot C$, follow CPOCO by CPOSL.
To compute $\text{DETERMINANT}(A)$, follow CPOCO by CPODI.
To compute $\text{INVERSE}(A)$, follow CPOCO by CPODI.

On Entry

A COMPLEX(LDA, N)
 the Hermitian matrix to be factored. Only the
 diagonal and upper triangle are used.

LDA INTEGER
 the leading dimension of the array A .

N INTEGER
 the order of the matrix A .

On Return

A an upper triangular matrix R so that $A = \text{CTRANS}(R) \cdot R$ where $\text{CTRANS}(R)$ is the conjugate transpose. The strict lower triangle is unaltered. If $\text{INFO} \neq 0$, the factorization is not complete.

RCOND REAL
 an estimate of the reciprocal condition of A .
 For the system $A \cdot X = B$, relative perturbations in A and B of size EPSILON may cause relative perturbations in X of size $\text{EPSILON}/\text{RCOND}$.
 If RCOND is so small that the logical expression
 $1.0 + \text{RCOND} \text{ .EQ. } 1.0$
 is true, then A may be singular to working precision. In particular, RCOND is zero if exact singularity is detected or the estimate underflows. If $\text{INFO} \neq 0$, RCOND is unchanged.

Z COMPLEX(N)
 a work vector whose contents are usually unimportant.
 If A is close to a singular matrix, then Z is an approximate null vector in the sense that

$\text{NORM}(A*Z) = \text{RCOND}*\text{NORM}(A)*\text{NORM}(Z)$.
If INFO .NE. 0 , Z is unchanged.

INFO INTEGER
 = 0 for normal return.
 = K signals an error condition. The leading minor
 of order K is not positive definite.

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
 Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CAXPY, CDOTC, CPOFA, CSSCAL, SCASUM

***REVISION HISTORY (YYMMDD)

780814 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900326 Removed duplicate information from DESCRIPTION section.
 (WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CPODI

```
      SUBROUTINE CPODI (A, LDA, N, DET, JOB)
***BEGIN PROLOGUE  CPODI
***PURPOSE  Compute the determinant and inverse of a certain complex
             Hermitian positive definite matrix using the factors
             computed by CPOCO, CPOFA, or CQRDC.
***LIBRARY  SLATEC (LINPACK)
***CATEGORY  D2D1B, D3D1B
***TYPE      COMPLEX (SPODI-S, DPODI-D, CPODI-C)
***KEYWORDS  DETERMINANT, INVERSE, LINEAR ALGEBRA, LINPACK, MATRIX,
             POSITIVE DEFINITE
***AUTHOR  Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CPODI computes the determinant and inverse of a certain complex Hermitian positive definite matrix (see below) using the factors computed by CPOCO, CPOFA or CQRDC.

On Entry

A COMPLEX(LDA, N)
 the output A from CPOCO or CPOFA
 or the output X from CQRDC.

LDA INTEGER
 the leading dimension of the array A .

N INTEGER
 the order of the matrix A .

JOB INTEGER
 = 11 both determinant and inverse.
 = 01 inverse only.
 = 10 determinant only.

On Return

A If CPOCO or CPOFA was used to factor A then
 CPODI produces the upper half of INVERSE(A) .
 If CQRDC was used to decompose X then
 CPODI produces the upper half of INVERSE(CTRANS(X)*X)
 where CTRANS(X) is the conjugate transpose.
 Elements of A below the diagonal are unchanged.
 If the units digit of JOB is zero, A is unchanged.

DET REAL(2)
 determinant of A or of CTRANS(X)*X if requested.
 Otherwise not referenced.
 Determinant = DET(1) * 10.0**DET(2)
 with 1.0 .LE. DET(1) .LT. 10.0
 or DET(1) .EQ. 0.0 .

Error Condition

a division by zero will occur if the input factor contains
a zero on the diagonal and the inverse is requested.
It will not occur if the subroutines are called correctly

```

        and if CPOCO or CPOFA has set INFO .EQ. 0 .

***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
                Stewart, LINPACK Users' Guide, SIAM, 1979.
***ROUTINES CALLED  CAXPY, CSCAL
***REVISION HISTORY  (YYMMDD)
    780814  DATE WRITTEN
    890831  Modified array declarations.  (WRB)
    890831  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    900326  Removed duplicate information from DESCRIPTION section.
            (WRB)
    920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE

```

CPOFA

```
SUBROUTINE CPOFA (A, LDA, N, INFO)
***BEGIN PROLOGUE  CPOFA
***PURPOSE  Factor a complex Hermitian positive definite matrix.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2D1B
***TYPE      COMPLEX (SPOFA-S, DPOFA-D, CPOFA-C)
***KEYWORDS  LINEAR ALGEBRA, LINPACK, MATRIX FACTORIZATION,
              POSITIVE DEFINITE
***AUTHOR   Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CPOFA factors a complex Hermitian positive definite matrix.

CPOFA is usually called by CPOCO, but it can be called directly with a saving in time if RCOND is not needed.
(Time for CPOCO) = (1 + 18/N)*(Time for CPOFA) .

On Entry

A COMPLEX(LDA, N)
 the Hermitian matrix to be factored. Only the
 diagonal and upper triangle are used.

LDA INTEGER
 the leading dimension of the array A .

N INTEGER
 the order of the matrix A .

On Return

A an upper triangular matrix R so that A =
 CTRANS(R)*R where CTRANS(R) is the conjugate
 transpose. The strict lower triangle is unaltered.
 If INFO .NE. 0 , the factorization is not complete.

INFO INTEGER
 = 0 for normal return.
 = K signals an error condition. The leading minor
 of order K is not positive definite.

```
***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
                Stewart, LINPACK Users' Guide, SIAM, 1979.
```

```
***ROUTINES CALLED  CDOTC
```

```
***REVISION HISTORY  (YYMMDD)
```

```
780814  DATE WRITTEN
890831  Modified array declarations.  (WRB)
890831  REVISION DATE from Version 3.2
891214  Prologue converted to Version 4.0 format.  (BAB)
900326  Removed duplicate information from DESCRIPTION section.
        (WRB)
920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

CPOFS

```
      SUBROUTINE CPOFS (A, LDA, N, V, ITASK, IND, WORK)
***BEGIN PROLOGUE  CPOFS
***PURPOSE  Solve a positive definite symmetric complex system of
            linear equations.
***LIBRARY   SLATEC
***CATEGORY  D2D1B
***TYPE      COMPLEX (SPOFS-S, DPOFS-D, CPOFS-C)
***KEYWORDS  HERMITIAN, LINEAR EQUATIONS, POSITIVE DEFINITE, SYMMETRIC
***AUTHOR  Voorhees, E. A., (LANL)
***DESCRIPTION
```

Subroutine CPOFS solves a positive definite symmetric NxN system of complex linear equations using LINPACK subroutines CPOCO and CPOSL. That is, if A is an NxN complex positive definite symmetric matrix and if X and B are complex N-vectors, then CPOFS solves the equation

$$A^*X=B.$$

Care should be taken not to use CPOFS with a non-Hermitian matrix.

The matrix A is first factored into upper and lower triangular matrices R and R-TRANPOSE. These factors are used to find the solution vector X. An approximate condition number is calculated to provide a rough estimate of the number of digits of accuracy in the computed solution.

If the equation $A^*X=B$ is to be solved for more than one vector B, the factoring of A does not need to be performed again and the option to only solve (ITASK .GT. 1) will be faster for the succeeding solutions. In this case, the contents of A, LDA, and N must not have been altered by the user following factorization (ITASK=1). IND will not be changed by CPOFS in this case.

Argument Description ***

A	COMPLEX(LDA,N) on entry, the doubly subscripted array with dimension (LDA,N) which contains the coefficient matrix. Only the upper triangle, including the diagonal, of the coefficient matrix need be entered and will subsequently be referenced and changed by the routine. on return, contains in its upper triangle an upper triangular matrix R such that $A = (R-TRANPOSE) * R$.
LDA	INTEGER the leading dimension of the array A. LDA must be greater than or equal to N. (terminal error message IND=-1)
N	INTEGER the order of the matrix A. N must be greater than or equal to 1. (terminal error message IND=-2)
V	COMPLEX(N) on entry the singly subscripted array(vector) of dimension N which contains the right hand side B of a system of simultaneous linear equations $A^*X=B$.

```

        on return, V contains the solution vector, X .
ITASK  INTEGER
        if ITASK = 1, the matrix A is factored and then the
           linear equation is solved.
        if ITASK .GT. 1, the equation is solved using the existing
           factored matrix A.
        if ITASK .LT. 1, then terminal error message IND=-3 is
           printed.
IND     INTEGER
        GT. 0   IND is a rough estimate of the number of digits
                of accuracy in the solution, X.
        LT. 0   see error message corresponding to IND below.
WORK    COMPLEX(N)
        a singly subscripted array of dimension at least N.

```

Error Messages Printed ***

```

IND=-1  terminal   N is greater than LDA.
IND=-2  terminal   N is less than 1.
IND=-3  terminal   ITASK is less than 1.
IND=-4  terminal   The matrix A is computationally singular or
                   is not positive definite. A solution
                   has not been computed.
IND=-10 warning    The solution has no apparent significance.
                   The solution may be inaccurate or the
                   matrix A may be poorly scaled.

```

NOTE- The above terminal(*fatal*) error messages are designed to be handled by XERMSG in which LEVEL=1 (recoverable) and IFLAG=2 . LEVEL=0 for warning error messages from XERMSG. Unless the user provides otherwise, an error message will be printed followed by an abort.

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CPOCO, CPOSL, R1MACH, XERMSG

***REVISION HISTORY (YYMMDD)

```

800516  DATE WRITTEN
890531  Changed all specific intrinsics to generic.  (WRB)
890831  Modified array declarations.  (WRB)
890831  REVISION DATE from Version 3.2
891214  Prologue converted to Version 4.0 format.  (BAB)
900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
900510  Convert XERRWV calls to XERMSG calls, cvt GOTO's to
        IF-THEN-ELSE.  (RWC)
920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE

```

CPOIR

```
      SUBROUTINE CPOIR (A, LDA, N, V, ITASK, IND, WORK)
***BEGIN PROLOGUE  CPOIR
***PURPOSE  Solve a positive definite Hermitian system of linear
            equations.  Iterative refinement is used to obtain an
            error estimate.
***LIBRARY    SLATEC
***CATEGORY  D2D1B
***TYPE       COMPLEX (SPOIR-S, CPOIR-C)
***KEYWORDS  HERMITIAN, LINEAR EQUATIONS, POSITIVE DEFINITE, SYMMETRIC
***AUTHOR  Voorhees, E. A., (LANL)
***DESCRIPTION
```

Subroutine CPOIR solves a complex positive definite Hermitian NxN system of single precision linear equations using LINPACK subroutines CPOFA and CPOSL. One pass of iterative refinement is used only to obtain an estimate of the accuracy. That is, if A is an NxN complex positive definite Hermitian matrix and if X and B are complex N-vectors, then CPOIR solves the equation

$$A \cdot X = B.$$

Care should be taken not to use CPOIR with a non-Hermitian matrix.

The matrix A is first factored into upper and lower triangular matrices R and R-TRANSPPOSE. These factors are used to calculate the solution, X. Then the residual vector is found and used to calculate an estimate of the relative error, IND. IND estimates the accuracy of the solution only when the input matrix and the right hand side are represented exactly in the computer and does not take into account any errors in the input data.

If the equation $A \cdot X = B$ is to be solved for more than one vector B, the factoring of A does not need to be performed again and the option to only solve (ITASK .GT. 1) will be faster for the succeeding solutions. In this case, the contents of A, LDA, N, and WORK must not have been altered by the user following factorization (ITASK=1). IND will not be changed by CPOIR in this case.

Argument Description ***

A	COMPLEX(LDA,N) the doubly subscripted array with dimension (LDA,N) which contains the coefficient matrix. Only the upper triangle, including the diagonal, of the coefficient matrix need be entered. A is not altered by the routine.
LDA	INTEGER the leading dimension of the array A. LDA must be greater than or equal to N. (terminal error message IND=-1)
N	INTEGER the order of the matrix A. N must be greater than or equal to one. (terminal error message IND=-2)

V COMPLEX(N)
 on entry, the singly subscripted array(vector) of dimension N which contains the right hand side B of a system of simultaneous linear equations $A \cdot X = B$.
 on return, V contains the solution vector, X .

ITASK INTEGER
 if ITASK = 1, the matrix A is factored and then the linear equation is solved.
 if ITASK .GT. 1, the equation is solved using the existing factored matrix A (stored in WORK).
 if ITASK .LT. 1, then terminal error IND=-3 is printed.

IND INTEGER
 GT. 0 IND is a rough estimate of the number of digits of accuracy in the solution, X. IND=75 means that the solution vector X is zero.
 LT. 0 see error message corresponding to IND below.

WORK COMPLEX(N*(N+1))
 a singly subscripted array of dimension at least N*(N+1).

Error Messages Printed ***

IND=-1	terminal	N is greater than LDA.
IND=-2	terminal	N is less than one.
IND=-3	terminal	ITASK is less than one.
IND=-4	terminal	The matrix A is computationally singular or is not positive definite. A solution has not been computed.
IND=-10	warning	The solution has no apparent significance. the solution may be inaccurate or the matrix a may be poorly scaled.

NOTE- the above terminal(*fatal*) error messages are designed to be handled by XERMSG in which LEVEL=1 (recoverable) and IFLAG=2 . LEVEL=0 for warning error messages from XERMSG. Unless the user provides otherwise, an error message will be printed followed by an abort.

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CCOPY, CPOFA, CPOSL, DCDOT, R1MACH, SCASUM, XERMSG

***REVISION HISTORY (YYMMDD)

800530 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900315 CALLs to XERROR changed to CALLs to XERMSG. (THJ)

900510 Convert XERRWV calls to XERMSG calls, cvt GOTO's to IF-THEN-ELSE. (RWC)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CPOSL

```
      SUBROUTINE CPOSL (A, LDA, N, B)
***BEGIN PROLOGUE  CPOSL
***PURPOSE  Solve the complex Hermitian positive definite linear system
            using the factors computed by CPOCO or CPOFA.
***LIBRARY    SLATEC (LINPACK)
***CATEGORY   D2D1B
***TYPE       COMPLEX (SPOSL-S, DPOSL-D, CPOSL-C)
***KEYWORDS   LINEAR ALGEBRA, LINPACK, MATRIX, POSITIVE DEFINITE, SOLVE
***AUTHOR    Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CPOSL solves the COMPLEX Hermitian positive definite system
 $A * X = B$
using the factors computed by CPOCO or CPOFA.

On Entry

A COMPLEX(LDA, N)
 the output from CPOCO or CPOFA.

LDA INTEGER
 the leading dimension of the array A .

N INTEGER
 the order of the matrix A .

B COMPLEX(N)
 the right hand side vector.

On Return

B the solution vector X .

Error Condition

A division by zero will occur if the input factor contains
a zero on the diagonal. Technically this indicates
singularity but it is usually caused by improper subroutine
arguments. It will not occur if the subroutines are called
correctly and INFO.EQ. 0 .

To compute $INVERSE(A) * C$ where C is a matrix
with P columns
 CALL CPOCO(A,LDA,N,RCOND,Z,INFO)
 IF (RCOND is too small .OR. INFO .NE. 0) GO TO ...
 DO 10 J = 1, P
 CALL CPOSL(A,LDA,N,C(1,J))
10 CONTINUE

```
***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
               Stewart, LINPACK Users' Guide, SIAM, 1979.
***ROUTINES CALLED  CAXPY, CDOTC
***REVISION HISTORY  (YYMMDD)
   780814  DATE WRITTEN
   890831  Modified array declarations.  (WRB)
   890831  REVISION DATE from Version 3.2
```

891214 Prologue converted to Version 4.0 format. (BAB)
900326 Removed duplicate information from DESCRIPTION section.
(WRB)
920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

CPPCO

```
SUBROUTINE CPPCO (AP, N, RCOND, Z, INFO)
***BEGIN PROLOGUE  CPPCO
***PURPOSE  Factor a complex Hermitian positive definite matrix stored
            in packed form and estimate the condition number of the
            matrix.
***LIBRARY    SLATEC (LINPACK)
***CATEGORY   D2D1B
***TYPE       COMPLEX (SPPCO-S, DPPCO-D, CPPCO-C)
***KEYWORDS   CONDITION NUMBER, LINEAR ALGEBRA, LINPACK,
            MATRIX FACTORIZATION, PACKED, POSITIVE DEFINITE
***AUTHOR    Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CPPCO factors a complex Hermitian positive definite matrix stored in packed form and estimates the condition of the matrix.

If RCOND is not needed, CPPFA is slightly faster.
To solve $A \cdot X = B$, follow CPPCO by CPPSL.
To compute $\text{INVERSE}(A) \cdot C$, follow CPPCO by CPPSL.
To compute $\text{DETERMINANT}(A)$, follow CPPCO by CPPDI.
To compute $\text{INVERSE}(A)$, follow CPPCO by CPPDI.

On Entry

AP COMPLEX (N*(N+1)/2)
 the packed form of a Hermitian matrix A . The
 columns of the upper triangle are stored sequentially
 in a one-dimensional array of length N*(N+1)/2 .
 See comments below for details.

N INTEGER
 the order of the matrix A .

On Return

AP an upper triangular matrix R , stored in packed
 form, so that $A = \text{CTRANS}(R) \cdot R$.
 If INFO .NE. 0 , the factorization is not complete.

RCOND REAL
 an estimate of the reciprocal condition of A .
 For the system $A \cdot X = B$, relative perturbations
 in A and B of size EPSILON may cause
 relative perturbations in X of size EPSILON/RCOND .
 If RCOND is so small that the logical expression
 $1.0 + \text{RCOND} \cdot \text{EQ} \cdot 1.0$
 is true, then A may be singular to working
 precision. In particular, RCOND is zero if
 exact singularity is detected or the estimate
 underflows. If INFO .NE. 0 , RCOND is unchanged.

Z COMPLEX(N)
 a work vector whose contents are usually unimportant.
 If A is singular to working precision, then Z is
 an approximate null vector in the sense that
 $\text{NORM}(A \cdot Z) = \text{RCOND} \cdot \text{NORM}(A) \cdot \text{NORM}(Z)$.

If INFO .NE. 0 , Z is unchanged.

INFO INTEGER
 = 0 for normal return.
 = K signals an error condition. The leading minor
 of order K is not positive definite.

Packed Storage

The following program segment will pack the upper triangle of a Hermitian matrix.

```
      K = 0
      DO 20 J = 1, N
        DO 10 I = 1, J
          K = K + 1
          AP(K) = A(I,J)
10      CONTINUE
20      CONTINUE
```

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CAXPY, CDOTC, CPPFA, CSSCAL, SCASUM

***REVISION HISTORY (YYMMDD)

780814 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900326 Removed duplicate information from DESCRIPTION section.
(WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CPPDI

```
      SUBROUTINE CPPDI (AP, N, DET, JOB)
***BEGIN PROLOGUE  CPPDI
***PURPOSE  Compute the determinant and inverse of a complex Hermitian
             positive definite matrix using factors from CPPCO or CPPFA.
***LIBRARY  SLATEC (LINPACK)
***CATEGORY  D2D1B, D3D1B
***TYPE      COMPLEX (SPPDI-S, DPPDI-D, CPPDI-C)
***KEYWORDS  DETERMINANT, INVERSE, LINEAR ALGEBRA, LINPACK, MATRIX,
             PACKED, POSITIVE DEFINITE
***AUTHOR  Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CPPDI computes the determinant and inverse
of a complex Hermitian positive definite matrix
using the factors computed by CPPCO or CPPFA .

On Entry

AP COMPLEX (N*(N+1)/2)
 the output from CPPCO or CPPFA.

N INTEGER
 the order of the matrix A .

JOB INTEGER
 = 11 both determinant and inverse.
 = 01 inverse only.
 = 10 determinant only.

On Return

AP the upper triangular half of the inverse .
 The strict lower triangle is unaltered.

DET REAL(2)
 determinant of original matrix if requested.
 Otherwise not referenced.
 Determinant = DET(1) * 10.0**DET(2)
 with 1.0 .LE. DET(1) .LT. 10.0
 or DET(1) .EQ. 0.0 .

Error Condition

A division by zero will occur if the input factor contains
a zero on the diagonal and the inverse is requested.
It will not occur if the subroutines are called correctly
and if CPOCO or CPOFA has set INFO .EQ. 0 .

```
***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
               Stewart, LINPACK Users' Guide, SIAM, 1979.
***ROUTINES CALLED  CAXPY, CSCAL
***REVISION HISTORY  (YYMMDD)
   780814  DATE WRITTEN
   890831  Modified array declarations.  (WRB)
   890831  REVISION DATE from Version 3.2
   891214  Prologue converted to Version 4.0 format.  (BAB)
```

900326 Removed duplicate information from DESCRIPTION section.
(WRB)
920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

CPPFA

```
      SUBROUTINE CPPFA (AP, N, INFO)
***BEGIN PROLOGUE  CPPFA
***PURPOSE  Factor a complex Hermitian positive definite matrix stored
             in packed form.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2D1B
***TYPE      COMPLEX (SPPFA-S, DPPFA-D, CPPFA-C)
***KEYWORDS  LINEAR ALGEBRA, LINPACK, MATRIX FACTORIZATION, PACKED,
             POSITIVE DEFINITE
***AUTHOR  Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CPPFA factors a complex Hermitian positive definite matrix stored in packed form.

CPPFA is usually called by CPPCO, but it can be called directly with a saving in time if RCOND is not needed.
(Time for CPPCO) = (1 + 18/N)*(Time for CPPFA) .

On Entry

AP COMPLEX (N*(N+1)/2)
 the packed form of a Hermitian matrix A . The columns of the upper triangle are stored sequentially in a one-dimensional array of length N*(N+1)/2 . See comments below for details.

N INTEGER
 the order of the matrix A .

On Return

AP an upper triangular matrix R , stored in packed form, so that $A = CTRANS(R)*R$.

INFO INTEGER
 = 0 for normal return.
 = K If the leading minor of order K is not positive definite.

Packed Storage

The following program segment will pack the upper triangle of a Hermitian matrix.

```
      K = 0
      DO 20 J = 1, N
        DO 10 I = 1, J
          K = K + 1
          AP(K) = A(I,J)
        10 CONTINUE
      20 CONTINUE
```

```
***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
               Stewart, LINPACK Users' Guide, SIAM, 1979.
```

```
***ROUTINES CALLED  CDOTC
***REVISION HISTORY  (YYMMDD)
  780814  DATE WRITTEN
  890831  Modified array declarations.  (WRB)
  890831  REVISION DATE from Version 3.2
  891214  Prologue converted to Version 4.0 format.  (BAB)
  900326  Removed duplicate information from DESCRIPTION section.
          (WRB)
  920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

CPPSL

```
      SUBROUTINE CPPSL (AP, N, B)
***BEGIN PROLOGUE  CPPSL
***PURPOSE  Solve the complex Hermitian positive definite system using
             the factors computed by CPPCO or CPPFA.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2D1B
***TYPE      COMPLEX (SPPSL-S, DPPSL-D, CPPSL-C)
***KEYWORDS  LINEAR ALGEBRA, LINPACK, MATRIX, PACKED,
             POSITIVE DEFINITE, SOLVE
***AUTHOR   Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CPPSL solves the complex Hermitian positive definite system
 $A * X = B$
using the factors computed by CPPCO or CPPFA.

On Entry

AP COMPLEX (N*(N+1)/2)
 the output from CPPCO or CPPFA.

N INTEGER
 the order of the matrix A .

B COMPLEX(N)
 the right hand side vector.

On Return

B the solution vector X .

Error Condition

A division by zero will occur if the input factor contains
a zero on the diagonal. Technically this indicates
singularity but it is usually caused by improper subroutine
arguments. It will not occur if the subroutines are called
correctly and INFO.EQ. 0 .

To compute INVERSE(A) * C where C is a matrix
with P columns
 CALL CPPCO(AP,N,RCOND,Z,INFO)
 IF (RCOND is too small .OR. INFO .NE. 0) GO TO ...
 DO 10 J = 1, P
 CALL CPPSL(AP,N,C(1,J))
10 CONTINUE

```
***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
               Stewart, LINPACK Users' Guide, SIAM, 1979.
***ROUTINES CALLED  CAXPY, CDOTC
***REVISION HISTORY  (YYMMDD)
   780814  DATE WRITTEN
   890831  Modified array declarations.  (WRB)
   890831  REVISION DATE from Version 3.2
   891214  Prologue converted to Version 4.0 format.  (BAB)
   900326  Removed duplicate information from DESCRIPTION section.
```

(WRB)
920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

CPQR79

```
SUBROUTINE CPQR79 (NDEG, COEFF, ROOT, IERR, WORK)
***BEGIN PROLOGUE  CPQR79
***PURPOSE  Find the zeros of a polynomial with complex coefficients.
***LIBRARY   SLATEC
***CATEGORY  F1A1B
***TYPE      COMPLEX (RPQR79-S, CPQR79-C)
***KEYWORDS  COMPLEX POLYNOMIAL, POLYNOMIAL ROOTS, POLYNOMIAL ZEROS
***AUTHOR  Vandevender, W. H., (SNLA)
***DESCRIPTION
```

Abstract

This routine computes all zeros of a polynomial of degree NDEG with complex coefficients by computing the eigenvalues of the companion matrix.

Description of Parameters

The user must dimension all arrays appearing in the call list
COEFF(NDEG+1), ROOT(NDEG), WORK(2*NDEG*(NDEG+1))

--Input--

NDEG degree of polynomial

COEFF COMPLEX coefficients in descending order. i.e.,
 $P(Z) = \text{COEFF}(1) \cdot (Z^{**NDEG}) + \text{COEFF}(NDEG) \cdot Z + \text{COEFF}(NDEG+1)$

WORK REAL work array of dimension at least 2*NDEG*(NDEG+1)

--Output--

ROOT COMPLEX vector of roots

IERR Output Error Code

- Normal Code
- 0 means the roots were computed.
- Abnormal Codes
- 1 more than 30 QR iterations on some eigenvalue of the
 companion matrix
- 2 COEFF(1)=0.0
- 3 NDEG is invalid (less than or equal to 0)

```
***REFERENCES  (NONE)
***ROUTINES CALLED  COMQR, XERMSG
***REVISION HISTORY  (YMMDD)
791201  DATE WRITTEN
890531  Changed all specific intrinsics to generic.  (WRB)
890531  REVISION DATE from Version 3.2
891214  Prologue converted to Version 4.0 format.  (BAB)
900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
900326  Removed duplicate information from DESCRIPTION section.
       (WRB)
911010  Code reworked and simplified.  (RWC and WRB)
END PROLOGUE
```

CPSI

```
      COMPLEX FUNCTION CPSI (ZIN)
***BEGIN PROLOGUE  CPSI
***PURPOSE  Compute the Psi (or Digamma) function.
***LIBRARY  SLATEC (FNLIB)
***CATEGORY  C7C
***TYPE      COMPLEX (PSI-S, DPSI-D, CPSI-C)
***KEYWORDS  DIGAMMA FUNCTION, FNLIB, PSI FUNCTION, SPECIAL FUNCTIONS
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION
```

PSI(X) calculates the psi (or digamma) function of X. PSI(X) is the logarithmic derivative of the gamma function of X.

```
***REFERENCES  (NONE)
***ROUTINES CALLED  CCOT, R1MACH, XERMSG
***REVISION HISTORY  (YYMMDD)
      780501  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890531  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
      900727  Added EXTERNAL statement.  (WRB)
      END PROLOGUE
```

CPTSL

```
      SUBROUTINE CPTSL (N, D, E, B)
***BEGIN PROLOGUE  CPTSL
***PURPOSE  Solve a positive definite tridiagonal linear system.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2D2A
***TYPE      COMPLEX (SPTSL-S, DPTSL-D, CPTSL-C)
***KEYWORDS  LINEAR ALGEBRA, LINPACK, MATRIX, POSITIVE DEFINITE, SOLVE,
              TRIDIAGONAL
***AUTHOR   Dongarra, J., (ANL)
***DESCRIPTION
```

CPTSL given a positive definite tridiagonal matrix and a right hand side will find the solution.

On Entry

N	INTEGER is the order of the tridiagonal matrix.
D	COMPLEX(N) is the diagonal of the tridiagonal matrix. On output D is destroyed.
E	COMPLEX(N) is the offdiagonal of the tridiagonal matrix. E(1) through E(N-1) should contain the offdiagonal.
B	COMPLEX(N) is the right hand side vector.

On Return

B	contains the solution.
---	------------------------

```
***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
                Stewart, LINPACK Users' Guide, SIAM, 1979.
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
    780814  DATE WRITTEN
    890505  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    900326  Removed duplicate information from DESCRIPTION section.
            (WRB)
    920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

CPZERO

```
SUBROUTINE CPZERO (IN, A, R, T, IFLG, S)
***BEGIN PROLOGUE  CPZERO
***PURPOSE  Find the zeros of a polynomial with complex coefficients.
***LIBRARY  SLATEC
***CATEGORY  F1A1B
***TYPE      COMPLEX (RPZERO-S, CPZERO-C)
***KEYWORDS  POLYNOMIAL ROOTS, POLYNOMIAL ZEROS, REAL ROOTS
***AUTHOR  Kahaner, D. K., (NBS)
***DESCRIPTION

    Find the zeros of the complex polynomial
      
$$P(Z) = A(1)Z^N + A(2)Z^{(N-1)} + \dots + A(N+1)$$


Input...
  IN = degree of P(Z)
  A = complex vector containing coefficients of P(Z),
      A(I) = coefficient of  $Z^{(N+1-i)}$ 
  R = N word complex vector containing initial estimates for zeros
      if these are known.
  T = 4(N+1) word array used for temporary storage
  IFLG = flag to indicate if initial estimates of
      zeros are input.
      If IFLG .EQ. 0, no estimates are input.
      If IFLG .NE. 0, the vector R contains estimates of
          the zeros
  ** WARNING ** If estimates are input, they must
      be separated, that is, distinct or
      not repeated.
  S = an N word array

Output...
  R(I) = Ith zero,
  S(I) = bound for R(I) .
  IFLG = error diagnostic
Error Diagnostics...
  If IFLG .EQ. 0 on return, all is well
  If IFLG .EQ. 1 on return, A(1)=0.0 or N=0 on input
  If IFLG .EQ. 2 on return, the program failed to converge
      after 25*N iterations.  Best current estimates of the
      zeros are in R(I).  Error bounds are not calculated.

***REFERENCES  (NONE)
***ROUTINES CALLED  CPEVL
***REVISION HISTORY  (YYMMDD)
  810223  DATE WRITTEN
  890531  Changed all specific intrinsics to generic.  (WRB)
  890531  REVISION DATE from Version 3.2
  891214  Prologue converted to Version 4.0 format.  (BAB)
END PROLOGUE
```

CQRDC

```
SUBROUTINE CQRDC (X, LDX, N, P, QRAUX, JPVT, WORK, JOB)
***BEGIN PROLOGUE  CQRDC
***PURPOSE  Use Householder transformations to compute the QR
             factorization of an N by P matrix.  Column pivoting is a
             users option.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D5
***TYPE      COMPLEX (SQRDC-S, DQORDC-D, CQRDC-C)
***KEYWORDS  LINEAR ALGEBRA, LINPACK, MATRIX, ORTHOGONAL TRIANGULAR,
             QR DECOMPOSITION
***AUTHOR   Stewart, G. W., (U. of Maryland)
***DESCRIPTION
```

CQRDC uses Householder transformations to compute the QR factorization of an N by P matrix X. Column pivoting based on the 2-norms of the reduced columns may be performed at the users option.

On Entry

X COMPLEX(LDX,P), where LDX .GE. N.
X contains the matrix whose decomposition is to be computed.

LDX INTEGER.
LDX is the leading dimension of the array X.

N INTEGER.
N is the number of rows of the matrix X.

P INTEGER.
P is the number of columns of the matrix X.

JVPT INTEGER(P).
JVPT contains integers that control the selection of the pivot columns. The K-th column X(K) of X is placed in one of three classes according to the value of JVPT(K).

 If JVPT(K) .GT. 0, then X(K) is an initial column.

 If JVPT(K) .EQ. 0, then X(K) is a free column.

 If JVPT(K) .LT. 0, then X(K) is a final column.

Before the decomposition is computed, initial columns are moved to the beginning of the array X and final columns to the end. Both initial and final columns are frozen in place during the computation and only free columns are moved. At the K-th stage of the reduction, if X(K) is occupied by a free column it is interchanged with the free column of largest reduced norm. JVPT is not referenced if JOB .EQ. 0.

WORK COMPLEX(P).
WORK is a work array. WORK is not referenced if
JOB .EQ. 0.

JOB INTEGER.
JOB is an integer that initiates column pivoting.
If JOB .EQ. 0, no pivoting is done.
If JOB .NE. 0, pivoting is done.

On Return

X X contains in its upper triangle the upper
 triangular matrix R of the QR factorization.
 Below its diagonal X contains information from
 which the unitary part of the decomposition
 can be recovered. Note that if pivoting has
 been requested, the decomposition is not that
 of the original matrix X but that of X
 with its columns permuted as described by JVPT.

QRAUX COMPLEX(P).
QRAUX contains further information required to recover
the unitary part of the decomposition.

JVPT JVPT(K) contains the index of the column of the
 original matrix that has been interchanged into
 the K-th column, if pivoting was requested.

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
 Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CAXPY, CDOTC, CSCAL, CSWAP, SCNRM2

***REVISION HISTORY (YMMDD)

780814 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900326 Removed duplicate information from DESCRIPTION section.
 (WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CQRSL

```

      SUBROUTINE CQRSL (X, LDX, N, K, QRAUX, Y, QY, QTY, B, RSD, XB,
+      JOB, INFO)
***BEGIN PROLOGUE  CQRSL
***PURPOSE  Apply the output of CQRDC to compute coordinate transfor-
             mations, projections, and least squares solutions.
***LIBRARY    SLATEC (LINPACK)
***CATEGORY   D9, D2C1
***TYPE       COMPLEX (SQRSL-S, DQRSL-D, CQRSL-C)
***KEYWORDS   LINEAR ALGEBRA, LINPACK, MATRIX, ORTHOGONAL TRIANGULAR,
             SOLVE
***AUTHOR    Stewart, G. W., (U. of Maryland)
***DESCRIPTION

```

CQRSL applies the output of CQRDC to compute coordinate transformations, projections, and least squares solutions. For K .LE. MIN(N,P), let XK be the matrix

$$XK = (X(JVPT(1)), X(JVPT(2)), \dots, X(JVPT(K)))$$

formed from columns JVPT(1), ... ,JVPT(K) of the original N x P matrix X that was input to CQRDC (if no pivoting was done, XK consists of the first K columns of X in their original order). CQRDC produces a factored unitary matrix Q and an upper triangular matrix R such that

$$XK = Q * \begin{pmatrix} R \\ 0 \end{pmatrix}$$

This information is contained in coded form in the arrays X and QRAUX.

On Entry

X	COMPLEX(LDX,P). X contains the output of CQRDC.
LDX	INTEGER. LDX is the leading dimension of the array X.
N	INTEGER. N is the number of rows of the matrix XK. It must have the same value as N in CQRDC.
K	INTEGER. K is the number of columns of the matrix XK. K must not be greater than (N,P), where P is the same as in the calling sequence to CQRDC.
QRAUX	COMPLEX(P). QRAUX contains the auxiliary output from CQRDC.
Y	COMPLEX(N) Y contains an N-vector that is to be manipulated by CQRSL.
JOB	INTEGER.

JOB specifies what is to be computed. JOB has the decimal expansion ABCDE, with the following meaning.

```

      If A .NE. 0, compute QY.
      If B,C,D, or E .NE. 0, compute QTY.
      If C .NE. 0, compute B.
      If D .NE. 0, compute RSD .
      If E .NE. 0, compute XB.

```

Note that a request to compute B, RSD, or XB automatically triggers the computation of QTY, for which an array must be provided in the calling sequence.

On Return

```

QY      COMPLEX(N).
        QY contains Q*Y, if its computation has been
        requested.

QTY     COMPLEX(N).
        QTY contains CTRANS(Q)*Y, if its computation has
        been requested. Here CTRANS(Q) is the conjugate
        transpose of the matrix Q.

B       COMPLEX(K)
        B contains the solution of the least squares problem

           minimize NORM2(Y - XK*B),

        if its computation has been requested. (Note that
        if pivoting was requested in CQRDC, the J-th
        component of B will be associated with column JVPT(J)
        of the original matrix X that was input into CQRDC.)

RSD     COMPLEX(N).
        RSD contains the least squares residual Y - XK*B,
        if its computation has been requested. RSD is
        also the orthogonal projection of Y onto the
        orthogonal complement of the column space of XK.

XB      COMPLEX(N).
        XB contains the least squares approximation XK*B,
        if its computation has been requested. XB is also
        the orthogonal projection of Y onto the column space
        of X.

INFO    INTEGER.
        INFO is zero unless the computation of B has
        been requested and R is exactly singular. In
        this case, INFO is the index of the first zero
        diagonal element of R and B is left unaltered.

```

The parameters QY, QTY, B, RSD, and XB are not referenced if their computation is not requested and in this case can be replaced by dummy variables in the calling program. To save storage, the user may in some cases use the same array for different parameters in the calling sequence. A frequently occurring example is when one wishes to compute

any of B, RSD, or XB and does not need Y or QTY. In this case one may identify Y, QTY, and one of B, RSD, or XB, while providing separate arrays for anything else that is to be computed. Thus the calling sequence

```
CALL CQRSL(X,LDX,N,K,QRAUX,Y,DUM,Y,B,Y,DUM,110,INFO)
```

will result in the computation of B and RSD, with RSD overwriting Y. More generally, each item in the following list contains groups of permissible identifications for a single calling sequence.

1. (Y,QTY,B) (RSD) (XB) (QY)
2. (Y,QTY,RSD) (B) (XB) (QY)
3. (Y,QTY,XB) (B) (RSD) (QY)
4. (Y,QY) (QTY,B) (RSD) (XB)
5. (Y,QY) (QTY,RSD) (B) (XB)
6. (Y,QY) (QTY,XB) (B) (RSD)

In any group the value returned in the array allocated to the group corresponds to the last member of the group.

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CAXPY, CCOPY, CDOTC

***REVISION HISTORY (YYMMDD)

780814 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900326 Removed duplicate information from DESCRIPTION section. (WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CROTG

```
      SUBROUTINE CROTG (CA, CB, C, S)
***BEGIN PROLOGUE  CROTG
***PURPOSE  Construct a Givens transformation.
***LIBRARY   SLATEC (BLAS)
***CATEGORY  D1B10
***TYPE      COMPLEX (SROTG-S, DROTG-D, CROTG-C)
***KEYWORDS  BLAS, GIVENS ROTATION, GIVENS TRANSFORMATION,
              LINEAR ALGEBRA, VECTOR
***AUTHOR   (UNKNOWN)
***DESCRIPTION

      Complex Givens transformation

      Construct the Givens transformation


$$G = \begin{pmatrix} C & S \\ -S & C \end{pmatrix}, \quad C^2 + \text{ABS}(S)^2 = 1,$$


      which zeros the second entry of the complex 2-vector (CA,CB)**T

      The quantity CA/ABS(CA)*NORM(CA,CB) overwrites CA in storage.

      Input:
          CA (Complex)
          CB (Complex)

      Output:
          CA (Complex)      CA/ABS(CA)*NORM(CA,CB)
          C  (Real)
          S  (Complex)

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      790101  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890531  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      END PROLOGUE
```

CSCAL

```
      SUBROUTINE CSCAL (N, CA, CX, INCX)
***BEGIN PROLOGUE  CSCAL
***PURPOSE  Multiply a vector by a constant.
***LIBRARY   SLATEC (BLAS)
***CATEGORY  D1A6
***TYPE      COMPLEX (SSCAL-S, DSCAL-D, CSCAL-C)
***KEYWORDS  BLAS, LINEAR ALGEBRA, SCALE, VECTOR
***AUTHOR   Lawson, C. L., (JPL)
            Hanson, R. J., (SNLA)
            Kincaid, D. R., (U. of Texas)
            Krogh, F. T., (JPL)
***DESCRIPTION

            B L A S  Subprogram
Description of Parameters

      --Input--
        N  number of elements in input vector(s)
        CA  complex scale factor
        CX  complex vector with N elements
        INCX  storage spacing between elements of CX

      --Output--
        CX  complex result (unchanged if N .LE. 0)

      Replace complex CX by complex CA*CX.
      For I = 0 to N-1, replace CX(IX+I*INCX) with CA*CX(IX+I*INCX),
      where IX = 1 if INCX .GE. 0, else IX = 1+(1-N)*INCX.

***REFERENCES  C. L. Lawson, R. J. Hanson, D. R. Kincaid and F. T.
            Krogh, Basic linear algebra subprograms for Fortran
            usage, Algorithm No. 539, Transactions on Mathematical
            Software 5, 3 (September 1979), pp. 308-323.
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      791001  DATE WRITTEN
      890831  Modified array declarations.  (WRB)
      890831  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      900821  Modified to correct problem with a negative increment.
            (WRB)
      920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

CSEVL

```
      FUNCTION CSEVL (X, CS, N)
***BEGIN PROLOGUE  CSEVL
***PURPOSE  Evaluate a Chebyshev series.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C3A2
***TYPE      SINGLE PRECISION (CSEVL-S, DCSEVL-D)
***KEYWORDS  CHEBYSHEV SERIES, FNLIB, SPECIAL FUNCTIONS
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION
```

Evaluate the N-term Chebyshev series CS at X. Adapted from
a method presented in the paper by Broucke referenced below.

Input Arguments --
X value at which the series is to be evaluated.
CS array of N terms of a Chebyshev series. In evaluating
 CS, only half the first coefficient is summed.
N number of terms in array CS.

```
***REFERENCES  R. Broucke, Ten subroutines for the manipulation of
               Chebyshev series, Algorithm 446, Communications of
               the A.C.M. 16, (1973) pp. 254-256.
               L. Fox and I. B. Parker, Chebyshev Polynomials in
               Numerical Analysis, Oxford University Press, 1968,
               page 56.
***ROUTINES CALLED  R1MACH, XERMSG
***REVISION HISTORY  (YYMMDD)
      770401  DATE WRITTEN
      890831  Modified array declarations.  (WRB)
      890831  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
      900329  Prologued revised extensively and code rewritten to allow
               X to be slightly outside interval (-1,+1).  (WRB)
      920501  Reformatted the REFERENCES section.  (WRB)
      END PROLOGUE
```

CSICO

```
SUBROUTINE CSICO (A, LDA, N, KPVT, RCOND, Z)
***BEGIN PROLOGUE  CSICO
***PURPOSE  Factor a complex symmetric matrix by elimination with
             symmetric pivoting and estimate the condition number of the
             matrix.
***LIBRARY    SLATEC (LINPACK)
***CATEGORY   D2C1
***TYPE       COMPLEX (SSICO-S, DSICO-D, CHICO-C, CSICO-C)
***KEYWORDS   CONDITION NUMBER, LINEAR ALGEBRA, LINPACK,
             MATRIX FACTORIZATION, SYMMETRIC
***AUTHOR    Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CSICO factors a complex symmetric matrix by elimination with symmetric pivoting and estimates the condition of the matrix.

If RCOND is not needed, CSIFA is slightly faster.
To solve $A \cdot X = B$, follow CSICO by CSISL.
To compute $\text{INVERSE}(A) \cdot C$, follow CSICO by CSISL.
To compute $\text{INVERSE}(A)$, follow CSICO by CSIDI.
To compute $\text{DETERMINANT}(A)$, follow CSICO by CSIDI.

On Entry

A COMPLEX(LDA, N)
 the symmetric matrix to be factored.
 Only the diagonal and upper triangle are used.

LDA INTEGER
 the leading dimension of the array A .

N INTEGER
 the order of the matrix A .

On Return

A a block diagonal matrix and the multipliers which
 were used to obtain it.
 The factorization can be written $A = U \cdot D \cdot \text{TRANS}(U)$
 where U is a product of permutation and unit
 upper triangular matrices, $\text{TRANS}(U)$ is the
 transpose of U, and D is block diagonal
 with 1 by 1 and 2 by 2 blocks.

KVPT INTEGER(N)
 an integer vector of pivot indices.

RCOND REAL
 an estimate of the reciprocal condition of A .
 For the system $A \cdot X = B$, relative perturbations
 in A and B of size EPSILON may cause
 relative perturbations in X of size $\text{EPSILON}/\text{RCOND}$.
 If RCOND is so small that the logical expression
 $1.0 + \text{RCOND} \cdot \text{EQ} \cdot 1.0$
 is true, then A may be singular to working
 precision. In particular, RCOND is zero if

exact singularity is detected or the estimate underflows.

Z COMPLEX(N)
 a work vector whose contents are usually unimportant.
 If A is close to a singular matrix, then Z is
 an approximate null vector in the sense that
 $\text{NORM}(A*Z) = \text{RCOND}*\text{NORM}(A)*\text{NORM}(Z)$.

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
 Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CAXPY, CDOTU, CSIFA, CSSCAL, SCASUM

***REVISION HISTORY (YYMMDD)

780814 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

891107 Corrected category and modified routine equivalence
list. (WRB)

891107 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900326 Removed duplicate information from DESCRIPTION section.
(WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CSIDI

```
SUBROUTINE CSIDI (A, LDA, N, KPVT, DET, WORK, JOB)
***BEGIN PROLOGUE  CSIDI
***PURPOSE  Compute the determinant and inverse of a complex symmetric
            matrix using the factors from CSIFA.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2C1, D3C1
***TYPE      COMPLEX (SSIDI-S, DSIDI-D, CHIDI-C, CSIDI-C)
***KEYWORDS  DETERMINANT, INVERSE, LINEAR ALGEBRA, LINPACK, MATRIX,
            SYMMETRIC
***AUTHOR    Bunch, J., (UCSD)
***DESCRIPTION
```

CSIDI computes the determinant and inverse
of a complex symmetric matrix using the factors from CSIFA.

On Entry

A COMPLEX(LDA,N)
 the output from CSIFA.

LDA INTEGER
 the leading dimension of the array A .

N INTEGER
 the order of the matrix A .

KPVT INTEGER(N)
 the pivot vector from CSIFA.

WORK COMPLEX(N)
 work vector. Contents destroyed.

JOB INTEGER
 JOB has the decimal expansion AB where
 If B .NE. 0, the inverse is computed,
 If A .NE. 0, the determinant is computed,

 For example, JOB = 11 gives both.

On Return

Variables not requested by JOB are not used.

A contains the upper triangle of the inverse of
 the original matrix. The strict lower triangle
 is never referenced.

DET COMPLEX(2)
 determinant of original matrix.
 Determinant = DET(1) * 10.0**DET(2)
 with 1.0 .LE. ABS(DET(1)) .LT. 10.0
 or DET(1) = 0.0.

Error Condition

A division by zero may occur if the inverse is requested

and CSICO has set RCOND .EQ. 0.0
or CSIFA has set INFO .NE. 0 .

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CAXPY, CCOPY, CDOTU, CSWAP

***REVISION HISTORY (YYMMDD)

780814 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

891107 Corrected category and modified routine equivalence
list. (WRB)

891107 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900326 Removed duplicate information from DESCRIPTION section.
(WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CSIFA

```
SUBROUTINE CSIFA (A, LDA, N, KPVT, INFO)
***BEGIN PROLOGUE  CSIFA
***PURPOSE  Factor a complex symmetric matrix by elimination with
             symmetric pivoting.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2C1
***TYPE      COMPLEX (SSIFA-S, DSIFA-D, CHIFA-C, CSIFA-C)
***KEYWORDS  LINEAR ALGEBRA, LINPACK, MATRIX FACTORIZATION, SYMMETRIC
***AUTHOR    Bunch, J., (UCSD)
***DESCRIPTION
```

CSIFA factors a complex symmetric matrix by elimination with symmetric pivoting.

To solve $A \cdot X = B$, follow CSIFA by CSISL.
To compute $INVERSE(A) \cdot C$, follow CSIFA by CSISL.
To compute $DETERMINANT(A)$, follow CSIFA by CSIDI.
To compute $INVERSE(A)$, follow CSIFA by CSIDI.

On Entry

A COMPLEX(LDA,N)
 the symmetric matrix to be factored.
 Only the diagonal and upper triangle are used.

LDA INTEGER
 the leading dimension of the array A .

N INTEGER
 the order of the matrix A .

On Return

A a block diagonal matrix and the multipliers which
 were used to obtain it.
 The factorization can be written $A = U \cdot D \cdot TRANS(U)$
 where U is a product of permutation and unit
 upper triangular matrices , TRANS(U) is the
 transpose of U , and D is block diagonal
 with 1 by 1 and 2 by 2 blocks.

KPVT INTEGER(N)
 an integer vector of pivot indices.

INFO INTEGER
 = 0 normal value.
 = K if the K-th pivot block is singular. This is
 not an error condition for this subroutine,
 but it does indicate that CSISL or CSIDI may
 divide by zero if called.

```
***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
                Stewart, LINPACK Users' Guide, SIAM, 1979.
***ROUTINES CALLED  CAXPY, CSWAP, ICAMAX
***REVISION HISTORY  (YYMMDD)
    780814  DATE WRITTEN
```

890531 Changed all specific intrinsics to generic. (WRB)
890831 Modified array declarations. (WRB)
891107 Corrected category and modified routine equivalence
list. (WRB)
891107 REVISION DATE from Version 3.2
891214 Prologue converted to Version 4.0 format. (BAB)
900326 Removed duplicate information from DESCRIPTION section.
(WRB)
920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

CSINH

```
      COMPLEX FUNCTION CSINH (Z)
***BEGIN PROLOGUE  CSINH
***PURPOSE  Compute the complex hyperbolic sine.
***LIBRARY  SLATEC (FNLIB)
***CATEGORY  C4C
***TYPE      COMPLEX (CSINH-C)
***KEYWORDS  ELEMENTARY FUNCTIONS, FNLIB, HYPERBOLIC SINE
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION

      CSINH(Z) calculates the complex hyperbolic sine of complex
      argument Z.  Z is in units of radians.

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      770401  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890531  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      END PROLOGUE
```

CSISL

```
      SUBROUTINE CSISL (A, LDA, N, KPVT, B)
***BEGIN PROLOGUE  CSISL
***PURPOSE  Solve a complex symmetric system using the factors obtained
            from CSIFA.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2C1
***TYPE      COMPLEX (SSISL-S, DSISL-D, CHISL-C, CSISL-C)
***KEYWORDS  LINEAR ALGEBRA, LINPACK, MATRIX, SOLVE, SYMMETRIC
***AUTHOR    Bunch, J., (UCSD)
***DESCRIPTION
```

CSISL solves the complex symmetric system
 $A * X = B$
using the factors computed by CSIFA.

On Entry

A COMPLEX(LDA,N)
 the output from CSIFA.

LDA INTEGER
 the leading dimension of the array A .

N INTEGER
 the order of the matrix A .

KVPT INTEGER(N)
 the pivot vector from CSIFA.

B COMPLEX(N)
 the right hand side vector.

On Return

B the solution vector X .

Error Condition

A division by zero may occur if CSICO has set RCOND .EQ. 0.0
or CSIFA has set INFO .NE. 0 .

To compute $INVERSE(A) * C$ where C is a matrix
with P columns

```
      CALL CSIFA(A,LDA,N,KVPT,INFO)
      If (INFO .NE. 0) GO TO ...
      DO 10 J = 1, P
        CALL CSISL(A,LDA,N,KVPT,C(1,j))
      10 CONTINUE
```

```
***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
               Stewart, LINPACK Users' Guide, SIAM, 1979.
***ROUTINES CALLED  CAXPY, CDOTU
***REVISION HISTORY  (YYMMDD)
      780814  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890831  Modified array declarations.  (WRB)
```

891107 Corrected category and modified routine equivalence
list. (WRB)
891107 REVISION DATE from Version 3.2
891214 Prologue converted to Version 4.0 format. (BAB)
900326 Removed duplicate information from DESCRIPTION section.
(WRB)
920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

CSPCO

```
SUBROUTINE CSPCO (AP, N, KPVT, RCOND, Z)
***BEGIN PROLOGUE  CSPCO
***PURPOSE  Factor a complex symmetric matrix stored in packed form
             by elimination with symmetric pivoting and estimate the
             condition number of the matrix.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2C1
***TYPE      COMPLEX (SSPCO-S, DSPCO-D, CHPCO-C, CSPCO-C)
***KEYWORDS  CONDITION NUMBER, LINEAR ALGEBRA, LINPACK,
             MATRIX FACTORIZATION, PACKED, SYMMETRIC
***AUTHOR   Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CSPCO factors a complex symmetric matrix stored in packed form by elimination with symmetric pivoting and estimates the condition of the matrix.

If RCOND is not needed, CSPFA is slightly faster.
To solve $A \cdot X = B$, follow CSPCO by CSPSL.
To compute $\text{INVERSE}(A) \cdot C$, follow CSPCO by CSPSL.
To compute $\text{INVERSE}(A)$, follow CSPCO by CSPDI.
To compute $\text{DETERMINANT}(A)$, follow CSPCO by CSPDI.

On Entry

AP COMPLEX (N*(N+1)/2)
 the packed form of a symmetric matrix A . The
 columns of the upper triangle are stored sequentially
 in a one-dimensional array of length N*(N+1)/2 .
 See comments below for details.

N INTEGER
 the order of the matrix A .

On Return

AP a block diagonal matrix and the multipliers which
 were used to obtain it stored in packed form.
 The factorization can be written $A = U \cdot D \cdot \text{TRANS}(U)$
 where U is a product of permutation and unit
 upper triangular matrices, TRANS(U) is the
 transpose of U, and D is block diagonal
 with 1 by 1 and 2 by 2 blocks.

KVPT INTEGER(N)
 an integer vector of pivot indices.

RCOND REAL
 an estimate of the reciprocal condition of A .
 For the system $A \cdot X = B$, relative perturbations
 in A and B of size EPSILON may cause
 relative perturbations in X of size EPSILON/RCOND .
 If RCOND is so small that the logical expression
 $1.0 + \text{RCOND} \cdot \text{EQ} \cdot 1.0$
 is true, then A may be singular to working
 precision. In particular, RCOND is zero if

exact singularity is detected or the estimate underflows.

Z COMPLEX(N)
 a work vector whose contents are usually unimportant.
 If A is close to a singular matrix, then Z is
 an approximate null vector in the sense that
 $\text{NORM}(A*Z) = \text{RCOND}*\text{NORM}(A)*\text{NORM}(Z)$.

Packed Storage

The following program segment will pack the upper triangle of a symmetric matrix.

```
      K = 0
      DO 20 J = 1, N
        DO 10 I = 1, J
          K = K + 1
          AP(K) = A(I,J)
10      CONTINUE
20 CONTINUE
```

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CAXPY, CDOTU, CSPFA, CSSCAL, SCASUM

***REVISION HISTORY (YYMMDD)

780814 DATE WRITTEN
890531 Changed all specific intrinsics to generic. (WRB)
890831 Modified array declarations. (WRB)
891107 Corrected category and modified routine equivalence list. (WRB)
891107 REVISION DATE from Version 3.2
891214 Prologue converted to Version 4.0 format. (BAB)
900326 Removed duplicate information from DESCRIPTION section. (WRB)
920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

CSPDI

```
      SUBROUTINE CSPDI (AP, N, KPVT, DET, WORK, JOB)
***BEGIN PROLOGUE  CSPDI
***PURPOSE  Compute the determinant and inverse of a complex symmetric
            matrix stored in packed form using the factors from CSPFA.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2C1, D3C1
***TYPE      COMPLEX (SSPDI-S, DSPDI-D, CHPDI-C, CSPDI-C)
***KEYWORDS  DETERMINANT, INVERSE, LINEAR ALGEBRA, LINPACK, MATRIX,
            PACKED, SYMMETRIC
***AUTHOR    Bunch, J., (UCSD)
***DESCRIPTION
```

CSPDI computes the determinant and inverse of a complex symmetric matrix using the factors from CSPFA, where the matrix is stored in packed form.

On Entry

AP COMPLEX (N*(N+1)/2)
 the output from CSPFA.

N INTEGER
 the order of the matrix A .

KPVT INTEGER(N)
 the pivot vector from CSPFA.

WORK COMPLEX(N)
 work vector. Contents ignored.

JOB INTEGER
 JOB has the decimal expansion AB where
 if B .NE. 0, the inverse is computed,
 if A .NE. 0, the determinant is computed.

 For example, JOB = 11 gives both.

On Return

Variables not requested by JOB are not used.

AP contains the upper triangle of the inverse of
 the original matrix, stored in packed form.
 The columns of the upper triangle are stored
 sequentially in a one-dimensional array.

DET COMPLEX(2)
 determinant of original matrix.
 Determinant = DET(1) * 10.0**DET(2)
 with 1.0 .LE. ABS(DET(1)) .LT. 10.0
 or DET(1) = 0.0.

Error Condition

A division by zero will occur if the inverse is requested
and CSPCO has set RCOND .EQ. 0.0

or CSPFA has set INFO .NE. 0 .

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CAXPY, CCOPY, CDOTU, CSWAP

***REVISION HISTORY (YYMMDD)

780814 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890831 Modified array declarations. (WRB)

891107 Corrected category and modified routine equivalence list. (WRB)

891107 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900326 Removed duplicate information from DESCRIPTION section. (WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CSPFA

```
SUBROUTINE CSPFA (AP, N, KPVT, INFO)
***BEGIN PROLOGUE  CSPFA
***PURPOSE  Factor a complex symmetric matrix stored in packed form by
             elimination with symmetric pivoting.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2C1
***TYPE      COMPLEX (SSPFA-S, DSPFA-D, CHPFA-C, CSPFA-C)
***KEYWORDS  LINEAR ALGEBRA, LINPACK, MATRIX FACTORIZATION, PACKED,
             SYMMETRIC
***AUTHOR    Bunch, J., (UCSD)
***DESCRIPTION
```

CSPFA factors a complex symmetric matrix stored in packed form by elimination with symmetric pivoting.

To solve $A \cdot X = B$, follow CSPFA by CSPSL.
To compute $\text{INVERSE}(A) \cdot C$, follow CSPFA by CSPSL.
To compute $\text{DETERMINANT}(A)$, follow CSPFA by CSPDI.
To compute $\text{INVERSE}(A)$, follow CSPFA by CSPDI.

On Entry

AP COMPLEX (N*(N+1)/2)
 the packed form of a symmetric matrix A . The
 columns of the upper triangle are stored sequentially
 in a one-dimensional array of length N*(N+1)/2 .
 See comments below for details.

N INTEGER
 the order of the matrix A .

On Return

AP a block diagonal matrix and the multipliers which
 were used to obtain it stored in packed form.
 The factorization can be written $A = U \cdot D \cdot \text{TRANS}(U)$
 where U is a product of permutation and unit
 upper triangular matrices , TRANS(U) is the
 transpose of U , and D is block diagonal
 with 1 by 1 and 2 by 2 blocks.

KPVT INTEGER(N)
 an integer vector of pivot indices.

INFO INTEGER
 = 0 normal value.
 = K if the K-th pivot block is singular. This is
 not an error condition for this subroutine,
 but it does indicate that CSPSL or CSPDI may
 divide by zero if called.

Packed Storage

The following program segment will pack the upper
triangle of a symmetric matrix.

```

      K = 0
      DO 20 J = 1, N
        DO 10 I = 1, J
          K = K + 1
          AP(K) = A(I,J)
10      CONTINUE
20 CONTINUE

```

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CAXPY, CSWAP, ICAMAX

***REVISION HISTORY (YYMMDD)

```

780814  DATE WRITTEN
890531  Changed all specific intrinsics to generic.  (WRB)
890831  Modified array declarations.  (WRB)
891107  Corrected category and modified routine equivalence
        list.  (WRB)
891107  REVISION DATE from Version 3.2
891214  Prologue converted to Version 4.0 format.  (BAB)
900326  Removed duplicate information from DESCRIPTION section.
        (WRB)
920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE

```

CSPSL

```
SUBROUTINE CSPSL (AP, N, KPVT, B)
***BEGIN PROLOGUE  CSPSL
***PURPOSE  Solve a complex symmetric system using the factors obtained
             from CSPFA.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2C1
***TYPE      COMPLEX (SSPSL-S, DSPSL-D, CHPSL-C, CSPSL-C)
***KEYWORDS  LINEAR ALGEBRA, LINPACK, MATRIX, PACKED, SOLVE, SYMMETRIC
***AUTHOR   Bunch, J., (UCSD)
***DESCRIPTION
```

CSISL solves the complex symmetric system
 $A * X = B$
using the factors computed by CSPFA.

On Entry

AP COMPLEX(N*(N+1)/2)
 the output from CSPFA.

N INTEGER
 the order of the matrix A .

KVPT INTEGER(N)
 the pivot vector from CSPFA.

B COMPLEX(N)
 the right hand side vector.

On Return

B the solution vector X .

Error Condition

A division by zero may occur if CSPCO has set RCOND .EQ. 0.0
or CSPFA has set INFO .NE. 0 .

To compute INVERSE(A) * C where C is a matrix
with P columns

```
CALL CSPFA(AP,N,KVPT,INFO)
IF (INFO .NE. 0) GO TO ...
DO 10 J = 1, P
    CALL CSPSL(AP,N,KVPT,C(1,J))
10 CONTINUE
```

```
***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
               Stewart, LINPACK Users' Guide, SIAM, 1979.
***ROUTINES CALLED  CAXPY, CDOTU
***REVISION HISTORY  (YYMMDD)
   780814  DATE WRITTEN
   890531  Changed all specific intrinsics to generic.  (WRB)
   890831  Modified array declarations.  (WRB)
   891107  Corrected category and modified routine equivalence
           list.  (WRB)
   891107  REVISION DATE from Version 3.2
```

891214 Prologue converted to Version 4.0 format. (BAB)
900326 Removed duplicate information from DESCRIPTION section.
(WRB)
920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

CSROT

```
SUBROUTINE CSROT (N, CX, INCX, CY, INCY, C, S)
***BEGIN PROLOGUE  CSROT
***PURPOSE  Apply a plane Givens rotation.
***LIBRARY   SLATEC (BLAS)
***CATEGORY  D1B10
***TYPE      COMPLEX (SROT-S, DROT-D, CSROT-C)
***KEYWORDS  BLAS, GIVENS ROTATION, GIVENS TRANSFORMATION,
              LINEAR ALGEBRA, PLANE ROTATION, VECTOR
***AUTHOR   Dongarra, J., (ANL)
***DESCRIPTION
```

CSROT applies the complex Givens rotation

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} C & S \\ -S & C \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix}$$

N times where for $I = 0, \dots, N-1$

$$\begin{aligned} X &= CX(LX+I*INCX) \\ Y &= CY(LY+I*INCY), \end{aligned}$$

where $LX = 1$ if $INCX \geq 0$, else $LX = 1+(1-N)*INCX$, and LY is defined in a similar way using $INCY$.

Argument Description

N	(integer)	number of elements in each vector
CX	(complex array)	beginning of one vector
INCX	(integer)	memory spacing of successive elements of vector CX
CY	(complex array)	beginning of the other vector
INCY	(integer)	memory spacing of successive elements of vector CY
C	(real)	cosine term of the rotation
S	(real)	sine term of the rotation.

```
***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
                Stewart, LINPACK Users' Guide, SIAM, 1979.
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
  810223  DATE WRITTEN
  890831  Modified array declarations.  (WRB)
  890831  REVISION DATE from Version 3.2
  891214  Prologue converted to Version 4.0 format.  (BAB)
  920310  Corrected definition of LX in DESCRIPTION.  (WRB)
  920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

CSSCAL

```
SUBROUTINE CSSCAL (N, SA, CX, INCX)
***BEGIN PROLOGUE  CSSCAL
***PURPOSE  Scale a complex vector.
***LIBRARY   SLATEC (BLAS)
***CATEGORY  D1A6
***TYPE      COMPLEX (CSSCAL-C)
***KEYWORDS  BLAS, LINEAR ALGEBRA, SCALE, VECTOR
***AUTHOR   Lawson, C. L., (JPL)
            Hanson, R. J., (SNLA)
            Kincaid, D. R., (U. of Texas)
            Krogh, F. T., (JPL)
***DESCRIPTION

            B L A S  Subprogram
Description of Parameters

--Input--
  N  number of elements in input vector(s)
  SA  single precision scale factor
  CX  complex vector with N elements
  INCX  storage spacing between elements of CX

--Output--
  CX  scaled result (unchanged if N .LE. 0)

Replace complex CX by (single precision SA) * (complex CX)
For I = 0 to N-1, replace CX(IX+I*INCX) with  SA * CX(IX+I*INCX),
where IX = 1 if INCX .GE. 0, else IX = 1+(1-N)*INCX.

***REFERENCES  C. L. Lawson, R. J. Hanson, D. R. Kincaid and F. T.
              Krogh, Basic linear algebra subprograms for Fortran
              usage, Algorithm No. 539, Transactions on Mathematical
              Software 5, 3 (September 1979), pp. 308-323.
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
  791001  DATE WRITTEN
  890831  Modified array declarations.  (WRB)
  890831  REVISION DATE from Version 3.2
  891214  Prologue converted to Version 4.0 format.  (BAB)
  900821  Modified to correct problem with a negative increment.
          (WRB)
  920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

CSVDC

```
      SUBROUTINE CSVDC (X, LDX, N, P, S, E, U, LDU, V, LDV, WORK, JOB,  
+      INFO)  
***BEGIN PROLOGUE  CSVDC  
***PURPOSE  Perform the singular value decomposition of a rectangular  
            matrix.  
***LIBRARY   SLATEC (LINPACK)  
***CATEGORY  D6  
***TYPE      COMPLEX (SSVDC-S, DSVDC-D, CSVDC-C)  
***KEYWORDS  LINEAR ALGEBRA, LINPACK, MATRIX,  
            SINGULAR VALUE DECOMPOSITION  
***AUTHOR   Stewart, G. W., (U. of Maryland)  
***DESCRIPTION
```

CSVDC is a subroutine to reduce a complex NxP matrix X by unitary transformations U and V to diagonal form. The diagonal elements S(I) are the singular values of X. The columns of U are the corresponding left singular vectors, and the columns of V the right singular vectors.

On Entry

X	COMPLEX(LDX,P), where LDX .GE. N. X contains the matrix whose singular value decomposition is to be computed. X is destroyed by CSVDC.						
LDX	INTEGER. LDX is the leading dimension of the array X.						
N	INTEGER. N is the number of rows of the matrix X.						
P	INTEGER. P is the number of columns of the matrix X.						
LDU	INTEGER. LDU is the leading dimension of the array U (see below).						
LDV	INTEGER. LDV is the leading dimension of the array V (see below).						
WORK	COMPLEX(N). WORK is a scratch array.						
JOB	INTEGER. JOB controls the computation of the singular vectors. It has the decimal expansion AB with the following meaning <table><tbody><tr><td>A .EQ. 0</td><td>Do not compute the left singular vectors.</td></tr><tr><td>A .EQ. 1</td><td>Return the N left singular vectors in U.</td></tr><tr><td>A .GE. 2</td><td>Return the first MIN(N,P)</td></tr></tbody></table>	A .EQ. 0	Do not compute the left singular vectors.	A .EQ. 1	Return the N left singular vectors in U.	A .GE. 2	Return the first MIN(N,P)
A .EQ. 0	Do not compute the left singular vectors.						
A .EQ. 1	Return the N left singular vectors in U.						
A .GE. 2	Return the first MIN(N,P)						

	left singular vectors in U.
B .EQ. 0	Do not compute the right singular vectors.
B .EQ. 1	Return the right singular vectors in V.

On Return

S COMPLEX(MM), where MM = MIN(N+1,P).
The first MIN(N,P) entries of S contain the singular values of X arranged in descending order of magnitude.

E COMPLEX(P).
E ordinarily contains zeros. However see the discussion of INFO for exceptions.

U COMPLEX(LDU,K), where LDU .GE. N. If JOBA .EQ. 1 then K .EQ. N. If JOBA .GE. 2 then K .EQ. MIN(N,P).
U contains the matrix of right singular vectors. U is not referenced if JOBA .EQ. 0. If N .LE. P or if JOBA .GT. 2, then U may be identified with X in the subroutine call.

V COMPLEX(LDV,P), where LDV .GE. P.
V contains the matrix of right singular vectors. V is not referenced if JOB .EQ. 0. If P .LE. N, then V may be identified with X in the subroutine call.

INFO INTEGER.
The singular values (and their corresponding singular vectors) S(INFO+1),S(INFO+2),...,S(M) are correct (here M=MIN(N,P)). Thus if INFO.EQ. 0, all the singular values and their vectors are correct. In any event, the matrix $B = CTRANS(U)*X*V$ is the bidiagonal matrix with the elements of S on its diagonal and the elements of E on its super-diagonal (CTRANS(U) is the conjugate-transpose of U). Thus the singular values of X and B are the same.

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CAXPY, CDOTC, CSCAL, CSROT, CSWAP, SCNRM2, SROTG

***REVISION HISTORY (YYMMDD)

790319 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890531 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900326 Removed duplicate information from DESCRIPTION section. (WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CSWAP

```
SUBROUTINE CSWAP (N, CX, INCX, CY, INCY)
***BEGIN PROLOGUE  CSWAP
***PURPOSE  Interchange two vectors.
***LIBRARY   SLATEC (BLAS)
***CATEGORY  D1A5
***TYPE      COMPLEX (SSWAP-S, DSWAP-D, CSWAP-C, ISWAP-I)
***KEYWORDS  BLAS, INTERCHANGE, LINEAR ALGEBRA, VECTOR
***AUTHOR    Lawson, C. L., (JPL)
              Hanson, R. J., (SNLA)
              Kincaid, D. R., (U. of Texas)
              Krogh, F. T., (JPL)
***DESCRIPTION

                B L A S  Subprogram
Description of Parameters

--Input--
  N  number of elements in input vector(s)
  CX  complex vector with N elements
  INCX  storage spacing between elements of CX
  CY  complex vector with N elements
  INCY  storage spacing between elements of CY

--Output--
  CX  input vector CY (unchanged if N .LE. 0)
  CY  input vector CX (unchanged if N .LE. 0)

Interchange complex CX and complex CY
For I = 0 to N-1, interchange CX(LX+I*INCX) and CY(LY+I*INCY),
where LX = 1 if INCX .GE. 0, else LX = 1+(1-N)*INCX, and LY is
defined in a similar way using INCY.

***REFERENCES  C. L. Lawson, R. J. Hanson, D. R. Kincaid and F. T.
              Krogh, Basic linear algebra subprograms for Fortran
              usage, Algorithm No. 539, Transactions on Mathematical
              Software 5, 3 (September 1979), pp. 308-323.
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
  791001  DATE WRITTEN
  890831  Modified array declarations.  (WRB)
  890831  REVISION DATE from Version 3.2
  891214  Prologue converted to Version 4.0 format.  (BAB)
  920310  Corrected definition of LX in DESCRIPTION.  (WRB)
  920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

CSYMM

```
      SUBROUTINE CSYMM (SIDE, UPLO, M, N, ALPHA, A, LDA, B, LBD, BETA,  
$      C, LDC)  
***BEGIN PROLOGUE  CSYMM  
***PURPOSE  Multiply a complex general matrix by a complex symmetric  
            matrix.  
***LIBRARY   SLATEC (BLAS)  
***CATEGORY  D1B6  
***TYPE      COMPLEX (SSYMM-S, DSYMM-D, CSYMM-C)  
***KEYWORDS  LEVEL 3 BLAS, LINEAR ALGEBRA  
***AUTHOR   Dongarra, J., (ANL)  
            Duff, I., (AERE)  
            Du Croz, J., (NAG)  
            Hammarling, S. (NAG)  
***DESCRIPTION
```

CSYMM performs one of the matrix-matrix operations

$$C := \alpha A^*B + \beta C,$$

or

$$C := \alpha B^*A + \beta C,$$

where α and β are scalars, A is a symmetric matrix and B and C are m by n matrices.

Parameters
=====

SIDE - CHARACTER*1.
On entry, **SIDE** specifies whether the symmetric matrix A appears on the left or right in the operation as follows:

$$\text{SIDE} = \text{'L'} \text{ or 'l'} \quad C := \alpha A^*B + \beta C,$$
$$\text{SIDE} = \text{'R'} \text{ or 'r'} \quad C := \alpha B^*A + \beta C,$$

Unchanged on exit.

UPLO - CHARACTER*1.
On entry, **UPLO** specifies whether the upper or lower triangular part of the symmetric matrix A is to be referenced as follows:

$$\text{UPLO} = \text{'U'} \text{ or 'u'} \quad \text{Only the upper triangular part of the symmetric matrix is to be referenced.}$$
$$\text{UPLO} = \text{'L'} \text{ or 'l'} \quad \text{Only the lower triangular part of the symmetric matrix is to be referenced.}$$

Unchanged on exit.

M - INTEGER.
On entry, **M** specifies the number of rows of the matrix C . M must be at least zero.
Unchanged on exit.

- N - INTEGER.
On entry, N specifies the number of columns of the matrix C.
N must be at least zero.
Unchanged on exit.
- ALPHA - COMPLEX .
On entry, ALPHA specifies the scalar alpha.
Unchanged on exit.
- A - COMPLEX array of DIMENSION (LDA, ka), where ka is m when SIDE = 'L' or 'l' and is n otherwise.
Before entry with SIDE = 'L' or 'l', the m by m part of the array A must contain the symmetric matrix, such that when UPLO = 'U' or 'u', the leading m by m upper triangular part of the array A must contain the upper triangular part of the symmetric matrix and the strictly lower triangular part of A is not referenced, and when UPLO = 'L' or 'l', the leading m by m lower triangular part of the array A must contain the lower triangular part of the symmetric matrix and the strictly upper triangular part of A is not referenced.
Before entry with SIDE = 'R' or 'r', the n by n part of the array A must contain the symmetric matrix, such that when UPLO = 'U' or 'u', the leading n by n upper triangular part of the array A must contain the upper triangular part of the symmetric matrix and the strictly lower triangular part of A is not referenced, and when UPLO = 'L' or 'l', the leading n by n lower triangular part of the array A must contain the lower triangular part of the symmetric matrix and the strictly upper triangular part of A is not referenced.
Unchanged on exit.
- LDA - INTEGER.
On entry, LDA specifies the first dimension of A as declared in the calling (sub) program. When SIDE = 'L' or 'l' then LDA must be at least max(1, m), otherwise LDA must be at least max(1, n).
Unchanged on exit.
- B - COMPLEX array of DIMENSION (LDB, n).
Before entry, the leading m by n part of the array B must contain the matrix B.
Unchanged on exit.
- LDB - INTEGER.
On entry, LDB specifies the first dimension of B as declared in the calling (sub) program. LDB must be at least max(1, m).
Unchanged on exit.
- BETA - COMPLEX .
On entry, BETA specifies the scalar beta. When BETA is supplied as zero then C need not be set on input.
Unchanged on exit.
- C - COMPLEX array of DIMENSION (LDC, n).
Before entry, the leading m by n part of the array C must contain the matrix C, except when beta is zero, in which

case C need not be set on entry.
On exit, the array C is overwritten by the m by n updated matrix.

LDC - INTEGER.
On entry, LDC specifies the first dimension of C as declared in the calling (sub) program. LDC must be at least max(1, m).
Unchanged on exit.

***REFERENCES Dongarra, J., Du Croz, J., Duff, I., and Hammarling, S.
A set of level 3 basic linear algebra subprograms.
ACM TOMS, Vol. 16, No. 1, pp. 1-17, March 1990.

***ROUTINES CALLED LSAME, XERBLA

***REVISION HISTORY (YYMMDD)

890208 DATE WRITTEN

910605 Modified to meet SLATEC prologue standards. Only comment lines were modified. (BKS)

END PROLOGUE

CSYR2K

```
      SUBROUTINE CSYR2K (UPLO, TRANS, N, K, ALPHA, A, LDA, B, LDB, BETA,
$      C, LDC)
***BEGIN PROLOGUE  CSYR2K
***PURPOSE  Perform symmetric rank 2k update of a complex symmetric
            matrix.
***LIBRARY    SLATEC (BLAS)
***CATEGORY   D1B6
***TYPE       COMPLEX (SSYR2-S, DSYR2-D, CSYR2-C, CSYR2K-C)
***KEYWORDS   LEVEL 3 BLAS, LINEAR ALGEBRA
***AUTHOR     Dongarra, J., (ANL)
              Duff, I., (AERE)
              Du Croz, J., (NAG)
              Hammarling, S. (NAG)
***DESCRIPTION
```

CSYR2K performs one of the symmetric rank 2k operations

$$C := \alpha A B' + \alpha B A' + \beta C,$$

or

$$C := \alpha A' B + \alpha B' A + \beta C,$$

where α and β are scalars, C is an n by n symmetric matrix and A and B are n by k matrices in the first case and k by n matrices in the second case.

Parameters
=====

UPLO - CHARACTER*1.
On entry, UPLO specifies whether the upper or lower triangular part of the array C is to be referenced as follows:

UPLO = 'U' or 'u' Only the upper triangular part of C is to be referenced.

UPLO = 'L' or 'l' Only the lower triangular part of C is to be referenced.

Unchanged on exit.

TRANS - CHARACTER*1.
On entry, TRANS specifies the operation to be performed as follows:

TRANS = 'N' or 'n' $C := \alpha A B' + \alpha B A' + \beta C.$

TRANS = 'T' or 't' $C := \alpha A' B + \alpha B' A + \beta C.$

Unchanged on exit.

N - INTEGER.

On entry, N specifies the order of the matrix C. N must be at least zero.
Unchanged on exit.

K - INTEGER.
On entry with TRANS = 'N' or 'n', K specifies the number of columns of the matrices A and B, and on entry with TRANS = 'T' or 't', K specifies the number of rows of the matrices A and B. K must be at least zero.
Unchanged on exit.

ALPHA - COMPLEX .
On entry, ALPHA specifies the scalar alpha.
Unchanged on exit.

A - COMPLEX array of DIMENSION (LDA, ka), where ka is k when TRANS = 'N' or 'n', and is n otherwise.
Before entry with TRANS = 'N' or 'n', the leading n by k part of the array A must contain the matrix A, otherwise the leading k by n part of the array A must contain the matrix A.
Unchanged on exit.

LDA - INTEGER.
On entry, LDA specifies the first dimension of A as declared in the calling (sub) program. When TRANS = 'N' or 'n' then LDA must be at least max(1, n), otherwise LDA must be at least max(1, k).
Unchanged on exit.

B - COMPLEX array of DIMENSION (LDB, kb), where kb is k when TRANS = 'N' or 'n', and is n otherwise.
Before entry with TRANS = 'N' or 'n', the leading n by k part of the array B must contain the matrix B, otherwise the leading k by n part of the array B must contain the matrix B.
Unchanged on exit.

LDB - INTEGER.
On entry, LDB specifies the first dimension of B as declared in the calling (sub) program. When TRANS = 'N' or 'n' then LDB must be at least max(1, n), otherwise LDB must be at least max(1, k).
Unchanged on exit.

BETA - COMPLEX .
On entry, BETA specifies the scalar beta.
Unchanged on exit.

C - COMPLEX array of DIMENSION (LDC, n).
Before entry with UPLO = 'U' or 'u', the leading n by n upper triangular part of the array C must contain the upper triangular part of the symmetric matrix and the strictly lower triangular part of C is not referenced. On exit, the upper triangular part of the array C is overwritten by the upper triangular part of the updated matrix.
Before entry with UPLO = 'L' or 'l', the leading n by n lower triangular part of the array C must contain the lower triangular part of the symmetric matrix and the strictly upper triangular part of C is not referenced. On exit, the

lower triangular part of the array C is overwritten by the lower triangular part of the updated matrix.

LDC - INTEGER.

On entry, LDC specifies the first dimension of C as declared in the calling (sub) program. LDC must be at least $\max(1, n)$.
Unchanged on exit.

***REFERENCES Dongarra, J., Du Croz, J., Duff, I., and Hammarling, S.
A set of level 3 basic linear algebra subprograms.
ACM TOMS, Vol. 16, No. 1, pp. 1-17, March 1990.

***ROUTINES CALLED LSAME, XERBLA

***REVISION HISTORY (YYMMDD)

890208 DATE WRITTEN

910605 Modified to meet SLATEC prologue standards. Only comment lines were modified. (BKS)

END PROLOGUE

CSYRK

```
      SUBROUTINE CSYRK (UPLO, TRANS, N, K, ALPHA, A, LDA, BETA, C, LDC)
***BEGIN PROLOGUE  CSYRK
***PURPOSE  Perform symmetric rank k update of a complex symmetric
            matrix.
***LIBRARY   SLATEC (BLAS)
***CATEGORY  D1B6
***TYPE      COMPLEX (SSYRK-S, DSYRK-D, CSYRK-C)
***KEYWORDS  LEVEL 3 BLAS, LINEAR ALGEBRA
***AUTHOR    Dongarra, J., (ANL)
            Duff, I., (AERE)
            Du Croz, J., (NAG)
            Hammarling, S. (NAG)
***DESCRIPTION
```

CSYRK performs one of the symmetric rank k operations

$$C := \alpha A A' + \beta C,$$

or

$$C := \alpha A' A + \beta C,$$

where α and β are scalars, C is an n by n symmetric matrix and A is an n by k matrix in the first case and a k by n matrix in the second case.

Parameters
=====

UPLO - CHARACTER*1.
On entry, UPLO specifies whether the upper or lower triangular part of the array C is to be referenced as follows:

UPLO = 'U' or 'u' Only the upper triangular part of C is to be referenced.

UPLO = 'L' or 'l' Only the lower triangular part of C is to be referenced.

Unchanged on exit.

TRANS - CHARACTER*1.
On entry, TRANS specifies the operation to be performed as follows:

TRANS = 'N' or 'n' $C := \alpha A A' + \beta C$.

TRANS = 'T' or 't' $C := \alpha A' A + \beta C$.

Unchanged on exit.

N - INTEGER.
On entry, N specifies the order of the matrix C . N must be at least zero.
Unchanged on exit.

K - INTEGER.
On entry with TRANS = 'N' or 'n', K specifies the number of columns of the matrix A, and on entry with TRANS = 'T' or 't', K specifies the number of rows of the matrix A. K must be at least zero.
Unchanged on exit.

ALPHA - COMPLEX .
On entry, ALPHA specifies the scalar alpha.
Unchanged on exit.

A - COMPLEX array of DIMENSION (LDA, ka), where ka is k when TRANS = 'N' or 'n', and is n otherwise.
Before entry with TRANS = 'N' or 'n', the leading n by k part of the array A must contain the matrix A, otherwise the leading k by n part of the array A must contain the matrix A.
Unchanged on exit.

LDA - INTEGER.
On entry, LDA specifies the first dimension of A as declared in the calling (sub) program. When TRANS = 'N' or 'n' then LDA must be at least max(1, n), otherwise LDA must be at least max(1, k).
Unchanged on exit.

BETA - COMPLEX .
On entry, BETA specifies the scalar beta.
Unchanged on exit.

C - COMPLEX array of DIMENSION (LDC, n).
Before entry with UPLO = 'U' or 'u', the leading n by n upper triangular part of the array C must contain the upper triangular part of the symmetric matrix and the strictly lower triangular part of C is not referenced. On exit, the upper triangular part of the array C is overwritten by the upper triangular part of the updated matrix.
Before entry with UPLO = 'L' or 'l', the leading n by n lower triangular part of the array C must contain the lower triangular part of the symmetric matrix and the strictly upper triangular part of C is not referenced. On exit, the lower triangular part of the array C is overwritten by the lower triangular part of the updated matrix.

LDC - INTEGER.
On entry, LDC specifies the first dimension of C as declared in the calling (sub) program. LDC must be at least max(1, n).
Unchanged on exit.

***REFERENCES Dongarra, J., Du Croz, J., Duff, I., and Hammarling, S.
A set of level 3 basic linear algebra subprograms.
ACM TOMS, Vol. 16, No. 1, pp. 1-17, March 1990.

***ROUTINES CALLED LSAME, XERBLA

***REVISION HISTORY (YYMMDD)
890208 DATE WRITTEN
910605 Modified to meet SLATEC prologue standards. Only comment lines were modified. (BKS)

END PROLOGUE

CTAN

```
      COMPLEX FUNCTION CTAN (Z)
***BEGIN PROLOGUE  CTAN
***PURPOSE  Compute the complex tangent.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  C4A
***TYPE      COMPLEX (CTAN-C)
***KEYWORDS  ELEMENTARY FUNCTIONS, FNLIB, TANGENT, TRIGONOMETRIC
***AUTHOR    Fullerton, W., (LANL)
***DESCRIPTION
```

CTAN(Z) calculates the complex trigonometric tangent of complex argument Z. Z is in units of radians.

```
***REFERENCES  (NONE)
***ROUTINES CALLED  R1MACH, XERCLR, XERMSG
***REVISION HISTORY  (YYMMDD)
    770401  DATE WRITTEN
    890531  Changed all specific intrinsics to generic.  (WRB)
    890531  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
    900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
END PROLOGUE
```

CTANH

```
      COMPLEX FUNCTION CTANH (Z)
***BEGIN PROLOGUE  CTANH
***PURPOSE  Compute the complex hyperbolic tangent.
***LIBRARY  SLATEC (FNLIB)
***CATEGORY  C4C
***TYPE      COMPLEX (CTANH-C)
***KEYWORDS  ELEMENTARY FUNCTIONS, FNLIB, HYPERBOLIC TANGENT
***AUTHOR  Fullerton, W., (LANL)
***DESCRIPTION

      CTANH(Z) calculates the complex hyperbolic tangent of complex
      argument Z.  Z is in units of radians.

***REFERENCES  (NONE)
***ROUTINES CALLED  CTAN
***REVISION HISTORY  (YYMMDD)
      770401  DATE WRITTEN
      861211  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      END PROLOGUE
```

CTBMV

```
      SUBROUTINE CTBMV (UPLO, TRANS, DIAG, N, K, A, LDA, X, INCX)
***BEGIN PROLOGUE  CTBMV
***PURPOSE  Multiply a complex vector by a complex triangular band
            matrix.
***LIBRARY   SLATEC (BLAS)
***CATEGORY  D1B4
***TYPE      COMPLEX (STBMV-S, DTBMV-D, CTBMV-C)
***KEYWORDS  LEVEL 2 BLAS, LINEAR ALGEBRA
***AUTHOR   Dongarra, J. J., (ANL)
            Du Croz, J., (NAG)
            Hammarling, S., (NAG)
            Hanson, R. J., (SNLA)
***DESCRIPTION
```

CTBMV performs one of the matrix-vector operations

$$x := A*x, \quad \text{or} \quad x := A'*x, \quad \text{or} \quad x := \text{conjg}(A')*x,$$

where x is an n element vector and A is an n by n unit, or non-unit, upper or lower triangular band matrix, with $(k + 1)$ diagonals.

Parameters
=====

UPLO - CHARACTER*1.
On entry, UPLO specifies whether the matrix is an upper or lower triangular matrix as follows:

UPLO = 'U' or 'u' A is an upper triangular matrix.

UPLO = 'L' or 'l' A is a lower triangular matrix.

Unchanged on exit.

TRANS - CHARACTER*1.
On entry, TRANS specifies the operation to be performed as follows:

TRANS = 'N' or 'n' $x := A*x$.

TRANS = 'T' or 't' $x := A'*x$.

TRANS = 'C' or 'c' $x := \text{conjg}(A')*x$.

Unchanged on exit.

DIAG - CHARACTER*1.
On entry, DIAG specifies whether or not A is unit triangular as follows:

DIAG = 'U' or 'u' A is assumed to be unit triangular.

DIAG = 'N' or 'n' A is not assumed to be unit triangular.

Unchanged on exit.

- N - INTEGER.
On entry, N specifies the order of the matrix A.
N must be at least zero.
Unchanged on exit.
- K - INTEGER.
On entry with UPLO = 'U' or 'u', K specifies the number of super-diagonals of the matrix A.
On entry with UPLO = 'L' or 'l', K specifies the number of sub-diagonals of the matrix A.
K must satisfy $0 \leq K$.
Unchanged on exit.
- A - COMPLEX array of DIMENSION (LDA, n).
Before entry with UPLO = 'U' or 'u', the leading (k + 1) by n part of the array A must contain the upper triangular band part of the matrix of coefficients, supplied column by column, with the leading diagonal of the matrix in row (k + 1) of the array, the first super-diagonal starting at position 2 in row k, and so on. The top left k by k triangle of the array A is not referenced.
The following program segment will transfer an upper triangular band matrix from conventional full matrix storage to band storage:

```

      DO 20, J = 1, N
        M = K + 1 - J
        DO 10, I = MAX( 1, J - K ), J
          A( M + I, J ) = matrix( I, J )
10      CONTINUE
20 CONTINUE

```

Before entry with UPLO = 'L' or 'l', the leading (k + 1) by n part of the array A must contain the lower triangular band part of the matrix of coefficients, supplied column by column, with the leading diagonal of the matrix in row 1 of the array, the first sub-diagonal starting at position 1 in row 2, and so on. The bottom right k by k triangle of the array A is not referenced.
The following program segment will transfer a lower triangular band matrix from conventional full matrix storage to band storage:

```

      DO 20, J = 1, N
        M = 1 - J
        DO 10, I = J, MIN( N, J + K )
          A( M + I, J ) = matrix( I, J )
10      CONTINUE
20 CONTINUE

```

Note that when DIAG = 'U' or 'u' the elements of the array A corresponding to the diagonal elements of the matrix are not referenced, but are assumed to be unity.
Unchanged on exit.

- LDA - INTEGER.
On entry, LDA specifies the first dimension of A as declared in the calling (sub) program. LDA must be at least (k + 1).

Unchanged on exit.

X - COMPLEX array of dimension at least
 (1 + (n - 1) * abs(INCX)).
 Before entry, the incremented array X must contain the n
 element vector x. On exit, X is overwritten with the
 transformed vector x.

INCX - INTEGER.
 On entry, INCX specifies the increment for the elements of
 X. INCX must not be zero.
 Unchanged on exit.

***REFERENCES Dongarra, J. J., Du Croz, J., Hammarling, S., and
 Hanson, R. J. An extended set of Fortran basic linear
 algebra subprograms. ACM TOMS, Vol. 14, No. 1,
 pp. 1-17, March 1988.

***ROUTINES CALLED LSAME, XERBLA

***REVISION HISTORY (YYMMDD)

861022 DATE WRITTEN

910605 Modified to meet SLATEC prologue standards. Only comment
 lines were modified. (BKS)

END PROLOGUE

CTBSV

```
SUBROUTINE CTBSV (UPLO, TRANS, DIAG, N, K, A, LDA, X, INCX)
***BEGIN PROLOGUE  CTBSV
***PURPOSE  Solve a complex triangular banded system of equations.
***LIBRARY   SLATEC (BLAS)
***CATEGORY  D1B4
***TYPE      COMPLEX (STBSV-S, DTBSV-D, CTBSV-C)
***KEYWORDS  LEVEL 2 BLAS, LINEAR ALGEBRA
***AUTHOR   Dongarra, J. J., (ANL)
            Du Croz, J., (NAG)
            Hammarling, S., (NAG)
            Hanson, R. J., (SNLA)
***DESCRIPTION
```

CTBSV solves one of the systems of equations

$$A*x = b, \quad \text{or} \quad A'*x = b, \quad \text{or} \quad \text{conjg}(A')*x = b,$$

where b and x are n element vectors and A is an n by n unit, or non-unit, upper or lower triangular band matrix, with (k + 1) diagonals.

No test for singularity or near-singularity is included in this routine. Such tests must be performed before calling this routine.

Parameters
=====

UPLO - CHARACTER*1.
On entry, UPLO specifies whether the matrix is an upper or lower triangular matrix as follows:

UPLO = 'U' or 'u' A is an upper triangular matrix.

UPLO = 'L' or 'l' A is a lower triangular matrix.

Unchanged on exit.

TRANS - CHARACTER*1.
On entry, TRANS specifies the equations to be solved as follows:

TRANS = 'N' or 'n' $A*x = b$.

TRANS = 'T' or 't' $A'*x = b$.

TRANS = 'C' or 'c' $\text{conjg}(A')*x = b$.

Unchanged on exit.

DIAG - CHARACTER*1.
On entry, DIAG specifies whether or not A is unit triangular as follows:

DIAG = 'U' or 'u' A is assumed to be unit triangular.

DIAG = 'N' or 'n' A is not assumed to be unit

triangular.

Unchanged on exit.

- N - INTEGER.
On entry, N specifies the order of the matrix A.
N must be at least zero.
Unchanged on exit.
- K - INTEGER.
On entry with UPLO = 'U' or 'u', K specifies the number of
super-diagonals of the matrix A.
On entry with UPLO = 'L' or 'l', K specifies the number of
sub-diagonals of the matrix A.
K must satisfy 0 .le. K.
Unchanged on exit.
- A - COMPLEX array of DIMENSION (LDA, n).
Before entry with UPLO = 'U' or 'u', the leading (k + 1)
by n part of the array A must contain the upper triangular
band part of the matrix of coefficients, supplied column by
column, with the leading diagonal of the matrix in row
(k + 1) of the array, the first super-diagonal starting at
position 2 in row k, and so on. The top left k by k triangle
of the array A is not referenced.
The following program segment will transfer an upper
triangular band matrix from conventional full matrix storage
to band storage:

```
      DO 20, J = 1, N
        M = K + 1 - J
        DO 10, I = MAX( 1, J - K ), J
          A( M + I, J ) = matrix( I, J )
10      CONTINUE
20      CONTINUE
```

Before entry with UPLO = 'L' or 'l', the leading (k + 1)
by n part of the array A must contain the lower triangular
band part of the matrix of coefficients, supplied column by
column, with the leading diagonal of the matrix in row 1 of
the array, the first sub-diagonal starting at position 1 in
row 2, and so on. The bottom right k by k triangle of the
array A is not referenced.
The following program segment will transfer a lower
triangular band matrix from conventional full matrix storage
to band storage:

```
      DO 20, J = 1, N
        M = 1 - J
        DO 10, I = J, MIN( N, J + K )
          A( M + I, J ) = matrix( I, J )
10      CONTINUE
20      CONTINUE
```

Note that when DIAG = 'U' or 'u' the elements of the array A
corresponding to the diagonal elements of the matrix are not
referenced, but are assumed to be unity.
Unchanged on exit.

- LDA - INTEGER.

On entry, LDA specifies the first dimension of A as declared in the calling (sub) program. LDA must be at least $(k + 1)$.
Unchanged on exit.

X - COMPLEX array of dimension at least $(1 + (n - 1) * \text{abs}(\text{INCX}))$.
Before entry, the incremented array X must contain the n element right-hand side vector b. On exit, X is overwritten with the solution vector x.

INCX - INTEGER.
On entry, INCX specifies the increment for the elements of X. INCX must not be zero.
Unchanged on exit.

***REFERENCES Dongarra, J. J., Du Croz, J., Hammarling, S., and Hanson, R. J. An extended set of Fortran basic linear algebra subprograms. ACM TOMS, Vol. 14, No. 1, pp. 1-17, March 1988.

***ROUTINES CALLED LSAME, XERBLA

***REVISION HISTORY (YYMMDD)

861022 DATE WRITTEN

910605 Modified to meet SLATEC prologue standards. Only comment lines were modified. (BKS)

END PROLOGUE

CTPMV

```
      SUBROUTINE CTPMV (UPLO, TRANS, DIAG, N, AP, X, INCX)
***BEGIN PROLOGUE  CTPMV
***PURPOSE  Perform one of the matrix-vector operations.
***LIBRARY   SLATEC (BLAS)
***CATEGORY  D1B4
***TYPE      COMPLEX (STPMV-S, DTPMV-D, CTPMV-C)
***KEYWORDS  LEVEL 2 BLAS, LINEAR ALGEBRA
***AUTHOR   Dongarra, J. J., (ANL)
             Du Croz, J., (NAG)
             Hammarling, S., (NAG)
             Hanson, R. J., (SNLA)
***DESCRIPTION
```

CTPMV performs one of the matrix-vector operations

$$x := A*x, \quad \text{or} \quad x := A'*x, \quad \text{or} \quad x := \text{conjg}(A')*x,$$

where x is an n element vector and A is an n by n unit, or non-unit, upper or lower triangular matrix, supplied in packed form.

Parameters
=====

UPLO - CHARACTER*1.
On entry, UPLO specifies whether the matrix is an upper or lower triangular matrix as follows:

UPLO = 'U' or 'u' A is an upper triangular matrix.

UPLO = 'L' or 'l' A is a lower triangular matrix.

Unchanged on exit.

TRANS - CHARACTER*1.
On entry, TRANS specifies the operation to be performed as follows:

TRANS = 'N' or 'n' $x := A*x$.

TRANS = 'T' or 't' $x := A'*x$.

TRANS = 'C' or 'c' $x := \text{conjg}(A')*x$.

Unchanged on exit.

DIAG - CHARACTER*1.
On entry, DIAG specifies whether or not A is unit triangular as follows:

DIAG = 'U' or 'u' A is assumed to be unit triangular.

DIAG = 'N' or 'n' A is not assumed to be unit triangular.

Unchanged on exit.

N - INTEGER.
 On entry, N specifies the order of the matrix A.
 N must be at least zero.
 Unchanged on exit.

AP - COMPLEX array of DIMENSION at least
 ((n*(n + 1))/2).
 Before entry with UPLO = 'U' or 'u', the array AP must
 contain the upper triangular matrix packed sequentially,
 column by column, so that AP(1) contains a(1, 1),
 AP(2) and AP(3) contain a(1, 2) and a(2, 2)
 respectively, and so on.
 Before entry with UPLO = 'L' or 'l', the array AP must
 contain the lower triangular matrix packed sequentially,
 column by column, so that AP(1) contains a(1, 1),
 AP(2) and AP(3) contain a(2, 1) and a(3, 1)
 respectively, and so on.
 Note that when DIAG = 'U' or 'u', the diagonal elements of
 A are not referenced, but are assumed to be unity.
 Unchanged on exit.

X - COMPLEX array of dimension at least
 (1 + (n - 1)*abs(INCX)).
 Before entry, the incremented array X must contain the n
 element vector x. On exit, X is overwritten with the
 transformed vector x.

INCX - INTEGER.
 On entry, INCX specifies the increment for the elements of
 X. INCX must not be zero.
 Unchanged on exit.

***REFERENCES Dongarra, J. J., Du Croz, J., Hammarling, S., and
 Hanson, R. J. An extended set of Fortran basic linear
 algebra subprograms. ACM TOMS, Vol. 14, No. 1,
 pp. 1-17, March 1988.

***ROUTINES CALLED LSAME, XERBLA

***REVISION HISTORY (YYMMDD)
 861022 DATE WRITTEN
 910605 Modified to meet SLATEC prologue standards. Only comment
 lines were modified. (BKS)
 END PROLOGUE

CTPSV

```
      SUBROUTINE CTPSV (UPLO, TRANS, DIAG, N, AP, X, INCX)
***BEGIN PROLOGUE  CTPSV
***PURPOSE  Solve one of the systems of equations.
***LIBRARY   SLATEC (BLAS)
***CATEGORY  D1B4
***TYPE      COMPLEX (STPSV-S, DTPSV-D, CTPSV-C)
***KEYWORDS  LEVEL 2 BLAS, LINEAR ALGEBRA
***AUTHOR    Dongarra, J. J., (ANL)
              Du Croz, J., (NAG)
              Hammarling, S., (NAG)
              Hanson, R. J., (SNLA)
***DESCRIPTION
```

CTPSV solves one of the systems of equations

$$A*x = b, \quad \text{or} \quad A'*x = b, \quad \text{or} \quad \text{conjg}(A')*x = b,$$

where b and x are n element vectors and A is an n by n unit, or non-unit, upper or lower triangular matrix, supplied in packed form.

No test for singularity or near-singularity is included in this routine. Such tests must be performed before calling this routine.

Parameters
=====

UPLO - CHARACTER*1.
On entry, UPLO specifies whether the matrix is an upper or lower triangular matrix as follows:

UPLO = 'U' or 'u' A is an upper triangular matrix.

UPLO = 'L' or 'l' A is a lower triangular matrix.

Unchanged on exit.

TRANS - CHARACTER*1.
On entry, TRANS specifies the equations to be solved as follows:

TRANS = 'N' or 'n' $A*x = b$.

TRANS = 'T' or 't' $A'*x = b$.

TRANS = 'C' or 'c' $\text{conjg}(A')*x = b$.

Unchanged on exit.

DIAG - CHARACTER*1.
On entry, DIAG specifies whether or not A is unit triangular as follows:

DIAG = 'U' or 'u' A is assumed to be unit triangular.

DIAG = 'N' or 'n' A is not assumed to be unit triangular.

Unchanged on exit.

N - INTEGER.
 On entry, N specifies the order of the matrix A.
 N must be at least zero.
 Unchanged on exit.

AP - COMPLEX array of DIMENSION at least
 ((n*(n + 1))/2).
 Before entry with UPLO = 'U' or 'u', the array AP must
 contain the upper triangular matrix packed sequentially,
 column by column, so that AP(1) contains a(1, 1),
 AP(2) and AP(3) contain a(1, 2) and a(2, 2)
 respectively, and so on.
 Before entry with UPLO = 'L' or 'l', the array AP must
 contain the lower triangular matrix packed sequentially,
 column by column, so that AP(1) contains a(1, 1),
 AP(2) and AP(3) contain a(2, 1) and a(3, 1)
 respectively, and so on.
 Note that when DIAG = 'U' or 'u', the diagonal elements of
 A are not referenced, but are assumed to be unity.
 Unchanged on exit.

X - COMPLEX array of dimension at least
 (1 + (n - 1)*abs(INCX)).
 Before entry, the incremented array X must contain the n
 element right-hand side vector b. On exit, X is overwritten
 with the solution vector x.

INCX - INTEGER.
 On entry, INCX specifies the increment for the elements of
 X. INCX must not be zero.
 Unchanged on exit.

***REFERENCES Dongarra, J. J., Du Croz, J., Hammarling, S., and
 Hanson, R. J. An extended set of Fortran basic linear
 algebra subprograms. ACM TOMS, Vol. 14, No. 1,
 pp. 1-17, March 1988.

***ROUTINES CALLED LSAME, XERBLA

***REVISION HISTORY (YYMMDD)
 861022 DATE WRITTEN
 910605 Modified to meet SLATEC prologue standards. Only comment
 lines were modified. (BKS)
 END PROLOGUE

CTRCO

```
SUBROUTINE CTCRCO (T, LDT, N, RCOND, Z, JOB)
***BEGIN PROLOGUE  CTCRCO
***PURPOSE  Estimate the condition number of a triangular matrix.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2C3
***TYPE      COMPLEX (STRCO-S, DTRCO-D, CTCRCO-C)
***KEYWORDS  CONDITION NUMBER, LINEAR ALGEBRA, LINPACK,
              TRIANGULAR MATRIX
***AUTHOR   Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CTRCO estimates the condition of a complex triangular matrix.

On Entry

T COMPLEX(LDT,N)
 T contains the triangular matrix. The zero
 elements of the matrix are not referenced, and
 the corresponding elements of the array can be
 used to store other information.

LDT INTEGER
 LDT is the leading dimension of the array T.

N INTEGER
 N is the order of the system.

JOB INTEGER
 = 0 T is lower triangular.
 = nonzero T is upper triangular.

On Return

RCOND REAL
 an estimate of the reciprocal condition of T .
 For the system $T \cdot X = B$, relative perturbations
 in T and B of size EPSILON may cause
 relative perturbations in X of size EPSILON/RCOND .
 If RCOND is so small that the logical expression
 $1.0 + RCOND \text{ .EQ. } 1.0$
 is true, then T may be singular to working
 precision. In particular, RCOND is zero if
 exact singularity is detected or the estimate
 underflows.

Z COMPLEX(N)
 a work vector whose contents are usually unimportant.
 If T is close to a singular matrix, then Z is
 an approximate null vector in the sense that
 $\text{NORM}(A \cdot Z) = \text{RCOND} \cdot \text{NORM}(A) \cdot \text{NORM}(Z)$.

```
***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
                Stewart, LINPACK Users' Guide, SIAM, 1979.
***ROUTINES CALLED  CAXPY, CSSCAL, SCASUM
***REVISION HISTORY  (YYMMDD)
    780814  DATE WRITTEN
```

890531 Changed all specific intrinsics to generic. (WRB)
890831 Modified array declarations. (WRB)
890831 REVISION DATE from Version 3.2
891214 Prologue converted to Version 4.0 format. (BAB)
900326 Removed duplicate information from DESCRIPTION section.
(WRB)
920501 Reformatted the REFERENCES section. (WRB)
END PROLOGUE

CTRDI

```
      SUBROUTINE CTRDI (T, LDT, N, DET, JOB, INFO)
***BEGIN PROLOGUE  CTRDI
***PURPOSE  Compute the determinant and inverse of a triangular matrix.
***LIBRARY   SLATEC (LINPACK)
***CATEGORY  D2C3, D3C3
***TYPE      COMPLEX (STRDI-S, DTRDI-D, CTRDI-C)
***KEYWORDS  DETERMINANT, INVERSE, LINEAR ALGEBRA, LINPACK,
              TRIANGULAR MATRIX
***AUTHOR  Moler, C. B., (U. of New Mexico)
***DESCRIPTION
```

CTRDI computes the determinant and inverse of a complex triangular matrix.

On Entry

T COMPLEX(LDT,N)
 T contains the triangular matrix. The zero elements of the matrix are not referenced, and the corresponding elements of the array can be used to store other information.

LDT INTEGER
 LDT is the leading dimension of the array T.

N INTEGER
 N is the order of the system.

JOB INTEGER
 = 010 no det, inverse of lower triangular.
 = 011 no det, inverse of upper triangular.
 = 100 det, no inverse.
 = 110 det, inverse of lower triangular.
 = 111 det, inverse of upper triangular.

On Return

T inverse of original matrix if requested.
 Otherwise unchanged.

DET COMPLEX(2)
 determinant of original matrix if requested.
 Otherwise not referenced.
 Determinant = DET(1) * 10.0**DET(2)
 with 1.0 .LE. CABS1(DET(1)) .LT. 10.0
 or DET(1) .EQ. 0.0 .

INFO INTEGER
 INFO contains zero if the system is nonsingular and the inverse is requested.
 Otherwise INFO contains the index of a zero diagonal element of T.

```
***REFERENCES  J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W.
                Stewart, LINPACK Users' Guide, SIAM, 1979.
```

```
***ROUTINES CALLED  CAXPY, CSCAL
***REVISION HISTORY  (YYMMDD)
  780814  DATE WRITTEN
  890831  Modified array declarations.  (WRB)
  890831  REVISION DATE from Version 3.2
  891214  Prologue converted to Version 4.0 format.  (BAB)
  900326  Removed duplicate information from DESCRIPTION section.
          (WRB)
  920501  Reformatted the REFERENCES section.  (WRB)
END PROLOGUE
```

CTRMM

```
      SUBROUTINE CTRMM (SIDE, UPLO, TRANSA, DIAG, M, N, ALPHA, A, LDA,  
$      B, LDB)  
***BEGIN PROLOGUE  CTRMM  
***PURPOSE  Multiply a complex general matrix by a complex triangular  
            matrix.  
***LIBRARY   SLATEC (BLAS)  
***CATEGORY  D1B6  
***TYPE      COMPLEX (STRMM-S, DTRMM-D, CTRMM-C)  
***KEYWORDS  LEVEL 3 BLAS, LINEAR ALGEBRA  
***AUTHOR    Dongarra, J., (ANL)  
            Duff, I., (AERE)  
            Du Croz, J., (NAG)  
            Hammarling, S. (NAG)  
***DESCRIPTION
```

CTRMM performs one of the matrix-matrix operations

$$B := \alpha * \text{op}(A) * B, \quad \text{or} \quad B := \alpha * B * \text{op}(A)$$

where α is a scalar, B is an m by n matrix, A is a unit, or non-unit, upper or lower triangular matrix and $\text{op}(A)$ is one of

$$\text{op}(A) = A \quad \text{or} \quad \text{op}(A) = A' \quad \text{or} \quad \text{op}(A) = \text{conjg}(A').$$

Parameters

=====

SIDE - CHARACTER*1.

On entry, SIDE specifies whether $\text{op}(A)$ multiplies B from the left or right as follows:

$$\text{SIDE} = 'L' \text{ or } 'l' \quad B := \alpha * \text{op}(A) * B.$$
$$\text{SIDE} = 'R' \text{ or } 'r' \quad B := \alpha * B * \text{op}(A).$$

Unchanged on exit.

UPLO - CHARACTER*1.

On entry, UPLO specifies whether the matrix A is an upper or lower triangular matrix as follows:

$$\text{UPLO} = 'U' \text{ or } 'u' \quad A \text{ is an upper triangular matrix.}$$
$$\text{UPLO} = 'L' \text{ or } 'l' \quad A \text{ is a lower triangular matrix.}$$

Unchanged on exit.

TRANSA - CHARACTER*1.

On entry, TRANSA specifies the form of $\text{op}(A)$ to be used in the matrix multiplication as follows:

$$\text{TRANSA} = 'N' \text{ or } 'n' \quad \text{op}(A) = A.$$
$$\text{TRANSA} = 'T' \text{ or } 't' \quad \text{op}(A) = A'.$$
$$\text{TRANSA} = 'C' \text{ or } 'c' \quad \text{op}(A) = \text{conjg}(A').$$

Unchanged on exit.

DIAG - CHARACTER*1.
On entry, DIAG specifies whether or not A is unit triangular as follows:

DIAG = 'U' or 'u' A is assumed to be unit triangular.

DIAG = 'N' or 'n' A is not assumed to be unit triangular.

Unchanged on exit.

M - INTEGER.
On entry, M specifies the number of rows of B. M must be at least zero.
Unchanged on exit.

N - INTEGER.
On entry, N specifies the number of columns of B. N must be at least zero.
Unchanged on exit.

ALPHA - COMPLEX .
On entry, ALPHA specifies the scalar alpha. When alpha is zero then A is not referenced and B need not be set before entry.
Unchanged on exit.

A - COMPLEX array of DIMENSION (LDA, k), where k is m when SIDE = 'L' or 'l' and is n when SIDE = 'R' or 'r'. Before entry with UPLO = 'U' or 'u', the leading k by k upper triangular part of the array A must contain the upper triangular matrix and the strictly lower triangular part of A is not referenced. Before entry with UPLO = 'L' or 'l', the leading k by k lower triangular part of the array A must contain the lower triangular matrix and the strictly upper triangular part of A is not referenced. Note that when DIAG = 'U' or 'u', the diagonal elements of A are not referenced either, but are assumed to be unity.
Unchanged on exit.

LDA - INTEGER.
On entry, LDA specifies the first dimension of A as declared in the calling (sub) program. When SIDE = 'L' or 'l' then LDA must be at least max(1, m), when SIDE = 'R' or 'r' then LDA must be at least max(1, n).
Unchanged on exit.

B - COMPLEX array of DIMENSION (LDB, n).
Before entry, the leading m by n part of the array B must contain the matrix B, and on exit is overwritten by the transformed matrix.

LDB - INTEGER.
On entry, LDB specifies the first dimension of B as declared in the calling (sub) program. LDB must be at least max(1, m).

Unchanged on exit.

```
***REFERENCES  Dongarra, J., Du Croz, J., Duff, I., and Hammarling, S.  
                A set of level 3 basic linear algebra subprograms.  
                ACM TOMS, Vol. 16, No. 1, pp. 1-17, March 1990.  
***ROUTINES CALLED  LSAME, XERBLA  
***REVISION HISTORY  (YYMMDD)  
    890208  DATE WRITTEN  
    910605  Modified to meet SLATEC prologue standards.  Only comment  
            lines were modified.  (BKS)  
END PROLOGUE
```

CTRMV

```
      SUBROUTINE CTRMV (UPLO, TRANS, DIAG, N, A, LDA, X, INCX)
***BEGIN PROLOGUE  CTRMV
***PURPOSE  Multiply a complex vector by a complex triangular matrix.
***LIBRARY   SLATEC (BLAS)
***CATEGORY  D1B4
***TYPE      COMPLEX (STRMV-S, DTRMV-D, CTRMV-C)
***KEYWORDS  LEVEL 2 BLAS, LINEAR ALGEBRA
***AUTHOR   Dongarra, J. J., (ANL)
             Du Croz, J., (NAG)
             Hammarling, S., (NAG)
             Hanson, R. J., (SNLA)
***DESCRIPTION
```

CTRMV performs one of the matrix-vector operations

$$x := A*x, \quad \text{or} \quad x := A'*x, \quad \text{or} \quad x := \text{conjg}(A')*x,$$

where x is an n element vector and A is an n by n unit, or non-unit, upper or lower triangular matrix.

Parameters
=====

UPLO - CHARACTER*1.
On entry, UPLO specifies whether the matrix is an upper or lower triangular matrix as follows:

UPLO = 'U' or 'u' A is an upper triangular matrix.

UPLO = 'L' or 'l' A is a lower triangular matrix.

Unchanged on exit.

TRANS - CHARACTER*1.
On entry, TRANS specifies the operation to be performed as follows:

TRANS = 'N' or 'n' x := A*x.

TRANS = 'T' or 't' x := A'*x.

TRANS = 'C' or 'c' x := conjg(A')*x.

Unchanged on exit.

DIAG - CHARACTER*1.
On entry, DIAG specifies whether or not A is unit triangular as follows:

DIAG = 'U' or 'u' A is assumed to be unit triangular.

DIAG = 'N' or 'n' A is not assumed to be unit triangular.

Unchanged on exit.

N - INTEGER.
 On entry, N specifies the order of the matrix A.
 N must be at least zero.
 Unchanged on exit.

A - COMPLEX array of DIMENSION (LDA, n).
 Before entry with UPLO = 'U' or 'u', the leading n by n
 upper triangular part of the array A must contain the upper
 triangular matrix and the strictly lower triangular part of
 A is not referenced.
 Before entry with UPLO = 'L' or 'l', the leading n by n
 lower triangular part of the array A must contain the lower
 triangular matrix and the strictly upper triangular part of
 A is not referenced.
 Note that when DIAG = 'U' or 'u', the diagonal elements of
 A are not referenced either, but are assumed to be unity.
 Unchanged on exit.

LDA - INTEGER.
 On entry, LDA specifies the first dimension of A as declared
 in the calling (sub) program. LDA must be at least
 max(1, n).
 Unchanged on exit.

X - COMPLEX array of dimension at least
 (1 + (n - 1) * abs(INCX)).
 Before entry, the incremented array X must contain the n
 element vector x. On exit, X is overwritten with the
 transformed vector x.

INCX - INTEGER.
 On entry, INCX specifies the increment for the elements of
 X. INCX must not be zero.
 Unchanged on exit.

***REFERENCES Dongarra, J. J., Du Croz, J., Hammarling, S., and
 Hanson, R. J. An extended set of Fortran basic linear
 algebra subprograms. ACM TOMS, Vol. 14, No. 1,
 pp. 1-17, March 1988.

***ROUTINES CALLED LSAME, XERBLA

***REVISION HISTORY (YYMMDD)
 861022 DATE WRITTEN
 910605 Modified to meet SLATEC prologue standards. Only comment
 lines were modified. (BKS)
 END PROLOGUE

CTRSL

```
SUBROUTINE CTRSL (T, LDT, N, B, JOB, INFO)
***BEGIN PROLOGUE  CTRSL
***PURPOSE  Solve a system of the form  $T^*X=B$  or  $CTRANS(T)^*X=B$ , where
             T is a triangular matrix. Here CTRANS(T) is the conjugate
             transpose.
***LIBRARY    SLATEC (LINPACK)
***CATEGORY   D2C3
***TYPE       COMPLEX (STRSL-S, DTRSL-D, CTRSL-C)
***KEYWORDS   LINEAR ALGEBRA, LINPACK, TRIANGULAR LINEAR SYSTEM,
             TRIANGULAR MATRIX
***AUTHOR     Stewart, G. W., (U. of Maryland)
***DESCRIPTION
```

CTRSL solves systems of the form

$$T * X = B$$

or

$$CTRANS(T) * X = B$$

where T is a triangular matrix of order N. Here CTRANS(T) denotes the conjugate transpose of the matrix T.

On Entry

T	COMPLEX(LDT,N) T contains the matrix of the system. The zero elements of the matrix are not referenced, and the corresponding elements of the array can be used to store other information.								
LDT	INTEGER LDT is the leading dimension of the array T.								
N	INTEGER N is the order of the system.								
B	COMPLEX(N). B contains the right hand side of the system.								
JOB	INTEGER JOB specifies what kind of system is to be solved. If JOB is <table><tbody><tr><td>00</td><td>solve $T^*X = B$, T lower triangular,</td></tr><tr><td>01</td><td>solve $T^*X = B$, T upper triangular,</td></tr><tr><td>10</td><td>solve $CTRANS(T)^*X = B$, T lower triangular,</td></tr><tr><td>11</td><td>solve $CTRANS(T)^*X = B$, T upper triangular.</td></tr></tbody></table>	00	solve $T^*X = B$, T lower triangular,	01	solve $T^*X = B$, T upper triangular,	10	solve $CTRANS(T)^*X = B$, T lower triangular,	11	solve $CTRANS(T)^*X = B$, T upper triangular.
00	solve $T^*X = B$, T lower triangular,								
01	solve $T^*X = B$, T upper triangular,								
10	solve $CTRANS(T)^*X = B$, T lower triangular,								
11	solve $CTRANS(T)^*X = B$, T upper triangular.								

On Return

B	B contains the solution, if INFO .EQ. 0. Otherwise B is unaltered.
INFO	INTEGER INFO contains zero if the system is nonsingular. Otherwise INFO contains the index of

the first zero diagonal element of T.

***REFERENCES J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, LINPACK Users' Guide, SIAM, 1979.

***ROUTINES CALLED CAXPY, CDOTC

***REVISION HISTORY (YYMMDD)

780814 DATE WRITTEN

890831 Modified array declarations. (WRB)

890831 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

900326 Removed duplicate information from DESCRIPTION section.
(WRB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

CTRSM

```
      SUBROUTINE CTRSM (SIDE, UPLO, TRANSA, DIAG, M, N, ALPHA, A, LDA,  
$      B, LDB)  
***BEGIN PROLOGUE  CTRSM  
***PURPOSE  Solve a complex triangular system of equations with  
             multiple right-hand sides.  
***LIBRARY   SLATEC (BLAS)  
***CATEGORY  D1B6  
***TYPE      COMPLEX (STRSM-S, DTRSM-D, CTRSM-C)  
***KEYWORDS  LEVEL 3 BLAS, LINEAR ALGEBRA  
***AUTHOR    Dongarra, J., (ANL)  
             Duff, I., (AERE)  
             Du Croz, J., (NAG)  
             Hammarling, S. (NAG)  
***DESCRIPTION
```

CTRSM solves one of the matrix equations

$$\text{op}(A) * X = \alpha * B, \quad \text{or} \quad X * \text{op}(A) = \alpha * B,$$

where α is a scalar, X and B are m by n matrices, A is a unit, or non-unit, upper or lower triangular matrix and $\text{op}(A)$ is one of

$$\text{op}(A) = A \quad \text{or} \quad \text{op}(A) = A' \quad \text{or} \quad \text{op}(A) = \text{conjg}(A').$$

The matrix X is overwritten on B .

Parameters
=====

SIDE - CHARACTER*1.

On entry, SIDE specifies whether $\text{op}(A)$ appears on the left or right of X as follows:

$$\text{SIDE} = 'L' \text{ or } 'l' \quad \text{op}(A) * X = \alpha * B.$$

$$\text{SIDE} = 'R' \text{ or } 'r' \quad X * \text{op}(A) = \alpha * B.$$

Unchanged on exit.

UPLO - CHARACTER*1.

On entry, UPLO specifies whether the matrix A is an upper or lower triangular matrix as follows:

$$\text{UPLO} = 'U' \text{ or } 'u' \quad A \text{ is an upper triangular matrix.}$$

$$\text{UPLO} = 'L' \text{ or } 'l' \quad A \text{ is a lower triangular matrix.}$$

Unchanged on exit.

TRANSA - CHARACTER*1.

On entry, TRANSA specifies the form of $\text{op}(A)$ to be used in the matrix multiplication as follows:

$$\text{TRANSA} = 'N' \text{ or } 'n' \quad \text{op}(A) = A.$$

$$\text{TRANSA} = 'T' \text{ or } 't' \quad \text{op}(A) = A'.$$

TRANSA = 'C' or 'c' op(A) = conjg(A').

Unchanged on exit.

DIAG - CHARACTER*1.

On entry, DIAG specifies whether or not A is unit triangular as follows:

DIAG = 'U' or 'u' A is assumed to be unit triangular.

DIAG = 'N' or 'n' A is not assumed to be unit triangular.

Unchanged on exit.

M - INTEGER.

On entry, M specifies the number of rows of B. M must be at least zero.

Unchanged on exit.

N - INTEGER.

On entry, N specifies the number of columns of B. N must be at least zero.

Unchanged on exit.

ALPHA - COMPLEX

On entry, ALPHA specifies the scalar alpha. When alpha is zero then A is not referenced and B need not be set before entry.

Unchanged on exit.

A - COMPLEX array of DIMENSION (LDA, k), where k is m when SIDE = 'L' or 'l' and is n when SIDE = 'R' or 'r'. Before entry with UPLO = 'U' or 'u', the leading k by k upper triangular part of the array A must contain the upper triangular matrix and the strictly lower triangular part of A is not referenced.

Before entry with UPLO = 'L' or 'l', the leading k by k lower triangular part of the array A must contain the lower triangular matrix and the strictly upper triangular part of A is not referenced.

Note that when DIAG = 'U' or 'u', the diagonal elements of A are not referenced either, but are assumed to be unity.

Unchanged on exit.

LDA - INTEGER.

On entry, LDA specifies the first dimension of A as declared in the calling (sub) program. When SIDE = 'L' or 'l' then LDA must be at least max(1, m), when SIDE = 'R' or 'r' then LDA must be at least max(1, n).

Unchanged on exit.

B - COMPLEX array of DIMENSION (LDB, n).

Before entry, the leading m by n part of the array B must contain the right-hand side matrix B, and on exit is overwritten by the solution matrix X.

LDB - INTEGER.

On entry, LDB specifies the first dimension of B as declared

in the calling (sub) program. LDB must be at least
max(1, m).
Unchanged on exit.

***REFERENCES Dongarra, J., Du Croz, J., Duff, I., and Hammarling, S.
A set of level 3 basic linear algebra subprograms.
ACM TOMS, Vol. 16, No. 1, pp. 1-17, March 1990.

***ROUTINES CALLED LSAME, XERBLA

***REVISION HISTORY (YYMMDD)

890208 DATE WRITTEN

910605 Modified to meet SLATEC prologue standards. Only comment
lines were modified. (BKS)

END PROLOGUE

CTRSV

```
      SUBROUTINE CTRSV (UPLO, TRANS, DIAG, N, A, LDA, X, INCX)
***BEGIN PROLOGUE  CTRSV
***PURPOSE  Solve a complex triangular system of equations.
***LIBRARY   SLATEC (BLAS)
***CATEGORY  D1B4
***TYPE      COMPLEX (STRSV-S, DTRSV-D, CTRSV-C)
***KEYWORDS  LEVEL 2 BLAS, LINEAR ALGEBRA
***AUTHOR   Dongarra, J. J., (ANL)
            Du Croz, J., (NAG)
            Hammarling, S., (NAG)
            Hanson, R. J., (SNLA)
***DESCRIPTION
```

CTRSV solves one of the systems of equations

$$A*x = b, \quad \text{or} \quad A'*x = b, \quad \text{or} \quad \text{conjg}(A')*x = b,$$

where b and x are n element vectors and A is an n by n unit, or non-unit, upper or lower triangular matrix.

No test for singularity or near-singularity is included in this routine. Such tests must be performed before calling this routine.

Parameters
=====

UPLO - CHARACTER*1.
On entry, UPLO specifies whether the matrix is an upper or lower triangular matrix as follows:

UPLO = 'U' or 'u' A is an upper triangular matrix.

UPLO = 'L' or 'l' A is a lower triangular matrix.

Unchanged on exit.

TRANS - CHARACTER*1.
On entry, TRANS specifies the equations to be solved as follows:

TRANS = 'N' or 'n' $A*x = b$.

TRANS = 'T' or 't' $A'*x = b$.

TRANS = 'C' or 'c' $\text{conjg}(A')*x = b$.

Unchanged on exit.

DIAG - CHARACTER*1.
On entry, DIAG specifies whether or not A is unit triangular as follows:

DIAG = 'U' or 'u' A is assumed to be unit triangular.

DIAG = 'N' or 'n' A is not assumed to be unit triangular.

Unchanged on exit.

N - INTEGER.
On entry, N specifies the order of the matrix A.
N must be at least zero.
Unchanged on exit.

A - COMPLEX array of DIMENSION (LDA, n).
Before entry with UPLO = 'U' or 'u', the leading n by n
upper triangular part of the array A must contain the upper
triangular matrix and the strictly lower triangular part of
A is not referenced.
Before entry with UPLO = 'L' or 'l', the leading n by n
lower triangular part of the array A must contain the lower
triangular matrix and the strictly upper triangular part of
A is not referenced.
Note that when DIAG = 'U' or 'u', the diagonal elements of
A are not referenced either, but are assumed to be unity.
Unchanged on exit.

LDA - INTEGER.
On entry, LDA specifies the first dimension of A as declared
in the calling (sub) program. LDA must be at least
max(1, n).
Unchanged on exit.

X - COMPLEX array of dimension at least
(1 + (n - 1) * abs(INCX)).
Before entry, the incremented array X must contain the n
element right-hand side vector b. On exit, X is overwritten
with the solution vector x.

INCX - INTEGER.
On entry, INCX specifies the increment for the elements of
X. INCX must not be zero.
Unchanged on exit.

***REFERENCES Dongarra, J. J., Du Croz, J., Hammarling, S., and
 Hanson, R. J. An extended set of Fortran basic linear
 algebra subprograms. ACM TOMS, Vol. 14, No. 1,
 pp. 1-17, March 1988.

***ROUTINES CALLED LSAME, XERBLA

***REVISION HISTORY (YYMMDD)
861022 DATE WRITTEN
910605 Modified to meet SLATEC prologue standards. Only comment
 lines were modified. (BKS)
END PROLOGUE

CV

```
      REAL FUNCTION CV (XVAL, NDATA, NCONST, NORD, NBKPT, BKPT, W)
***BEGIN PROLOGUE  CV
***PURPOSE  Evaluate the variance function of the curve obtained
             by the constrained B-spline fitting subprogram FC.
***LIBRARY   SLATEC
***CATEGORY  L7A3
***TYPE      SINGLE PRECISION (CV-S, DCV-D)
***KEYWORDS  ANALYSIS OF COVARIANCE, B-SPLINE,
             CONSTRAINED LEAST SQUARES, CURVE FITTING
***AUTHOR   Hanson, R. J., (SNLA)
***DESCRIPTION
```

CV() is a companion function subprogram for FC(). The documentation for FC() has complete usage instructions.

CV() is used to evaluate the variance function of the curve obtained by the constrained B-spline fitting subprogram, FC(). The variance function defines the square of the probable error of the fitted curve at any point, XVAL. One can use the square root of this variance function to determine a probable error band around the fitted curve.

CV() is used after a call to FC(). MODE, an input variable to FC(), is used to indicate if the variance function is desired. In order to use CV(), MODE must equal 2 or 4 on input to FC(). MODE is also used as an output flag from FC(). Check to make sure that MODE = 0 after calling FC(), indicating a successful constrained curve fit. The array SDDATA, as input to FC(), must also be defined with the standard deviation or uncertainty of the Y values to use CV().

To evaluate the variance function after calling FC() as stated above, use CV() as shown here

```
      VAR=CV(XVAL,NDATA,NCONST,NORD,NBKPT,BKPT,W)
```

The variance function is given by

```
      VAR=(transpose of B(XVAL))*C*B(XVAL)/MAX(NDATA-N,1)
```

where $N = NBKPT - NORD$.

The vector B(XVAL) is the B-spline basis function values at $X=XVAL$. The covariance matrix, C, of the solution coefficients accounts only for the least squares equations and the explicitly stated equality constraints. This fact must be considered when interpreting the variance function from a data fitting problem that has inequality constraints on the fitted curve.

All the variables in the calling sequence for CV() are used in FC() except the variable XVAL. Do not change the values of these variables between the call to FC() and the use of CV().

The following is a brief description of the variables

XVAL The point where the variance is desired.

NDATA The number of discrete (X,Y) pairs for which FC()
 calculated a piece-wise polynomial curve.

NCONST The number of conditions that constrained the B-spline in
 FC().

NORD The order of the B-spline used in FC().
 The value of NORD must satisfy $1 < \text{NORD} < 20$.

 (The order of the spline is one more than the degree of
 the piece-wise polynomial defined on each interval. This
 is consistent with the B-spline package convention. For
 example, NORD=4 when we are using piece-wise cubics.)

NBKPT The number of knots in the array BKPT(*).
 The value of NBKPT must satisfy $\text{NBKPT} \geq 2 * \text{NORD}$.

BKPT(*) The real array of knots. Normally the problem data
 interval will be included between the limits BKPT(NORD)
 and BKPT(NBKPT-NORD+1). The additional end knots
 BKPT(I), I=1,...,NORD-1 and I=NBKPT-NORD+2,...,NBKPT, are
 required by FC() to compute the functions used to fit
 the data.

W(*) Real work array as used in FC(). See FC() for the
 required length of W(*). The contents of W(*) must not
 be modified by the user if the variance function is
 desired.

***REFERENCES R. J. Hanson, Constrained least squares curve fitting
 to discrete data using B-splines, a users guide,
 Report SAND78-1291, Sandia Laboratories, December
 1978.

***ROUTINES CALLED BSPLVN, SDOT

***REVISION HISTORY (YYMMDD)

780801 DATE WRITTEN

890531 Changed all specific intrinsics to generic. (WRB)

890531 REVISION DATE from Version 3.2

891214 Prologue converted to Version 4.0 format. (BAB)

920501 Reformatted the REFERENCES section. (WRB)

END PROLOGUE

D1MACH

```
DOUBLE PRECISION FUNCTION D1MACH (I)
***BEGIN PROLOGUE  D1MACH
***PURPOSE  Return floating point machine dependent constants.
***LIBRARY    SLATEC
***CATEGORY   R1
***TYPE       DOUBLE PRECISION (R1MACH-S, D1MACH-D)
***KEYWORDS   MACHINE CONSTANTS
***AUTHOR    Fox, P. A., (Bell Labs)
              Hall, A. D., (Bell Labs)
              Schryer, N. L., (Bell Labs)
***DESCRIPTION
```

D1MACH can be used to obtain machine-dependent parameters for the local machine environment. It is a function subprogram with one (input) argument, and can be referenced as follows:

$$D = D1MACH(I)$$

where $I=1,\dots,5$. The (output) value of D above is determined by the (input) value of I . The results for various values of I are discussed below.

```
D1MACH( 1) = B**(EMIN-1), the smallest positive magnitude.
D1MACH( 2) = B**EMAX*(1 - B**(-T)), the largest magnitude.
D1MACH( 3) = B**(-T), the smallest relative spacing.
D1MACH( 4) = B**(1-T), the largest relative spacing.
D1MACH( 5) = LOG10(B)
```

Assume double precision numbers are represented in the T -digit, base- B form

$$\text{sign} (B**E)*((X(1)/B) + \dots + (X(T)/B**T))$$

where $0 \leq X(I) < B$ for $I=1,\dots,T$, $0 < X(1)$, and $EMIN \leq E \leq EMAX$.

The values of B , T , $EMIN$ and $EMAX$ are provided in I1MACH as follows:

```
I1MACH(10) = B, the base.
I1MACH(14) = T, the number of base-B digits.
I1MACH(15) = EMIN, the smallest exponent E.
I1MACH(16) = EMAX, the largest exponent E.
```

To alter this function for a particular environment, the desired set of DATA statements should be activated by removing the C from column 1. Also, the values of D1MACH(1) - D1MACH(4) should be checked for consistency with the local operating system.

```
***REFERENCES  P. A. Fox, A. D. Hall and N. L. Schryer, Framework for
               a portable library, ACM Transactions on Mathematical
               Software 4, 2 (June 1978), pp. 177-188.
***ROUTINES CALLED  XERMSG
***REVISION HISTORY  (YYMMDD)
    750101  DATE WRITTEN
    890213  REVISION DATE from Version 3.2
    891214  Prologue converted to Version 4.0 format.  (BAB)
```

900315 CALLs to XERROR changed to CALLs to XERMSG. (THJ)
900618 Added DEC RISC constants. (WRB)
900723 Added IBM RS 6000 constants. (WRB)
900911 Added SUN 386i constants. (WRB)
910710 Added HP 730 constants. (SMR)
911114 Added Convex IEEE constants. (WRB)
920121 Added SUN -r8 compiler option constants. (WRB)
920229 Added Touchstone Delta i860 constants. (WRB)
920501 Reformatted the REFERENCES section. (WRB)
920625 Added CONVEX -p8 and -pd8 compiler option constants.
(BKS, WRB)
930201 Added DEC Alpha and SGI constants. (RWC and WRB)
END PROLOGUE

D9PAK

```
      DOUBLE PRECISION FUNCTION D9PAK (Y, N)
***BEGIN PROLOGUE  D9PAK
***PURPOSE  Pack a base 2 exponent into a floating point number.
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  A6B
***TYPE      DOUBLE PRECISION (R9PAK-S, D9PAK-D)
***KEYWORDS  FNLIB, PACK
***AUTHOR   Fullerton, W., (LANL)
***DESCRIPTION
```

Pack a base 2 exponent into floating point number X. This routine is almost the inverse of D9UPAK. It is not exactly the inverse, because $\text{ABS}(X)$ need not be between 0.5 and 1.0. If both D9PAK and $2.d0**N$ were known to be in range we could compute

$$D9PAK = X * 2.0d0**N$$

```
***REFERENCES  (NONE)
***ROUTINES CALLED  D1MACH, D9UPAK, I1MACH, XERMSG
***REVISION HISTORY  (YYMMDD)
  790801  DATE WRITTEN
  890531  Changed all specific intrinsics to generic.  (WRB)
  890911  Removed unnecessary intrinsics.  (WRB)
  891009  Corrected error when XERROR called.  (WRB)
  891009  REVISION DATE from Version 3.2
  891214  Prologue converted to Version 4.0 format.  (BAB)
  900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
  901009  Routine used I1MACH(7) where it should use I1MACH(10),
          Corrected (RWC)
END PROLOGUE
```

D9UPAK

```
      SUBROUTINE D9UPAK (X, Y, N)
***BEGIN PROLOGUE  D9UPAK
***PURPOSE  Unpack a floating point number X so that  $X = Y*2**N$ .
***LIBRARY   SLATEC (FNLIB)
***CATEGORY  A6B
***TYPE      DOUBLE PRECISION (R9UPAK-S, D9UPAK-D)
***KEYWORDS  FNLIB, UNPACK
***AUTHOR   Fullerton, W., (LANL)
***DESCRIPTION

      Unpack a floating point number X so that  $X = Y*2.0**N$ , where
      0.5 .LE. ABS(Y) .LT. 1.0.

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      780701  DATE WRITTEN
      890531  Changed all specific intrinsics to generic.  (WRB)
      890531  REVISION DATE from Version 3.2
      891214  Prologue converted to Version 4.0 format.  (BAB)
      900820  Corrected code to find Y between 0.5 and 1.0 rather than
              between 0.05 and 1.0.  (WRB)
      END PROLOGUE
```

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government thereof, and shall not be used for advertising or product endorsement purposes. (C) Copyright 1996 The Regents of the University of California. All rights reserved.

Structural Keyword Index

Keyword	Description
<u>entire</u>	This entire document.
<u>title</u>	The name of this document.
<u>scope</u>	Topics covered in SLATEC2.
<u>availability</u>	Machines on which these routines run.
<u>who</u>	Who to contact for assistance.
<u>introduction</u>	Brief overview of SLATEC2; and other SLATEC documentation.
<u>index</u>	This structural keyword index.
<u>date</u>	The latest revision date for SLATEC2.
<u>revisions</u>	Revision history of this document.

In addition, the name of every subroutine described in SLATEC2 is the keyword and link for retrieving its description. Included are:

Routine Name	Gams Cat.	Function Performed
<u>AAAAAA</u>	z	documentation
<u>ACOSH</u>	c	elementary-functions, special-functions
<u>AI</u>	c	elementary-functions, special-functions
<u>AIE</u>	c	elementary-functions, special-functions
<u>ALBETA</u>	c	elementary-functions, special-functions
<u>ALGAMS</u>	c	elementary-functions, special-functions
<u>ALI</u>	c	elementary-functions, special-functions
<u>ALNGAM</u>	c	elementary-functions, special-functions
<u>ALNREL</u>	c	elementary-functions, special-functions
<u>ASINH</u>	c	elementary-functions, special-functions
<u>ATANH</u>	c	elementary-functions, special-functions
<u>AVINT</u>	h2	quadrature, definite-integrals
<u>BAKVEC</u>	eispack	
<u>BALANC</u>	eispack	
<u>BALBAK</u>	eispack	
<u>BANDR</u>	eispack	
<u>BANDV</u>	eispack	
<u>BESI</u>	c	elementary-functions, special-functions
<u>BESIO</u>	c	elementary-functions, special-functions
<u>BESIOE</u>	c	elementary-functions, special-functions
<u>BESI1</u>	c	elementary-functions, special-functions
<u>BESI1E</u>	c	elementary-functions, special-functions
<u>BESJ</u>	c	elementary-functions, special-functions
<u>BESJO</u>	c	elementary-functions, special-functions
<u>BESJ1</u>	c	elementary-functions, special-functions
<u>BESK</u>	c	elementary-functions, special-functions
<u>BESK0</u>	c	elementary-functions, special-functions
<u>BESK0E</u>	c	elementary-functions, special-functions
<u>BESK1</u>	c	elementary-functions, special-functions
<u>BESK1E</u>	c	elementary-functions, special-functions
<u>BESKES</u>	c	elementary-functions, special-functions

<u>BESKS</u>	c	elementary-functions, special-functions
<u>BESY</u>	c	elementary-functions, special-functions
<u>BESY0</u>	c	elementary-functions, special-functions
<u>BESY1</u>	c	elementary-functions, special-functions
<u>BETA</u>	c	elementary-functions, special-functions
<u>BETAI</u>	c	elementary-functions, special-functions
<u>BFQAD</u>	e	interpolation
<u>BI</u>	c	elementary-functions, special-functions
<u>BIE</u>	c	elementary-functions, special-functions
<u>BINOM</u>	c	elementary-functions, special-functions
<u>BINT4</u>	e	interpolation
<u>BINTK</u>	e	interpolation
<u>BISECT</u>	eispack	
<u>BLKTRI</u>	i2	partial-differential-equations
<u>BNDACC</u>	d9	overdetermined-systems, least-squares
<u>BNDSOL</u>	d9	overdetermined-systems, least-squares
<u>BQR</u>	eispack	
<u>BSKIN</u>	c	elementary-functions, special-functions
<u>BSPDOC</u>	z	documentation
<u>BSPDR</u>	e	interpolation
<u>BSPEV</u>	e	interpolation
<u>BSPPP</u>	e	interpolation
<u>BSPVD</u>	e	interpolation
<u>BSPVN</u>	e	interpolation
<u>BSQAD</u>	e	interpolation
<u>BVALU</u>	e	interpolation
<u>BVSUP</u>	i1	ordinary-differential-equations
<u>COLGMC</u>	c	elementary-functions, special-functions
<u>CACOS</u>	c	elementary-functions, special-functions
<u>CACOSH</u>	c	elementary-functions, special-functions
<u>CAIRY</u>	c	elementary-functions, special-functions
<u>CARG</u>	c	elementary-functions, special-functions
<u>CASIN</u>	c	elementary-functions, special-functions
<u>CASINH</u>	c	elementary-functions, special-functions
<u>CATAN</u>	c	elementary-functions, special-functions
<u>CATAN2</u>	c	elementary-functions, special-functions
<u>CATANH</u>	c	elementary-functions, special-functions
<u>CAXPY</u>	d1a	vector-operations
<u>CBABK2</u>	eispack	
<u>CBAL</u>	eispack	
<u>CBESH</u>	c	elementary-functions, special-functions
<u>CBESI</u>	c	elementary-functions, special-functions
<u>CBESJ</u>	c	elementary-functions, special-functions
<u>CBESK</u>	c	elementary-functions, special-functions
<u>CBESY</u>	c	elementary-functions, special-functions
<u>CBETA</u>	c	elementary-functions, special-functions
<u>CBIRY</u>	c	elementary-functions, special-functions
<u>CBLKTR</u>	i2	partial-differential-equations
<u>CBRT</u>	c	elementary-functions, special-functions
<u>CCBRT</u>	c	elementary-functions, special-functions
<u>CCHDC</u>	linpack	cholesky-operations
<u>CCHDD</u>	linpack	cholesky-operations
<u>CCHEX</u>	linpack	cholesky-operations
<u>CCHUD</u>	linpack	cholesky-operations
<u>CCOPY</u>	d1a	vector-operations
<u>CCOSH</u>	c	elementary-functions, special-functions
<u>CCOT</u>	c	elementary-functions, special-functions
<u>CDCDOT</u>	d1a	vector-operations
<u>CDOTC</u>	d1a	vector-operations
<u>CDOTU</u>	d1a	vector-operations

<u>CDRIV1</u>	i1	ordinary-differential-equations
<u>CDRIV2</u>	i1	ordinary-differential-equations
<u>CDRIV3</u>	i1	ordinary-differential-equations
<u>CEXPRL</u>	c	elementary-functions, special-functions
<u>CFFTB1</u>	j1	fast-fourier-transforms
<u>CFFTF1</u>	j1	fast-fourier-transforms
<u>CFFTI1</u>	j1	fast-fourier-transforms
<u>CG</u>	eispack	
<u>CGAMMA</u>	c	elementary-functions, special-functions
<u>CGAMR</u>	c	elementary-functions, special-functions
<u>CGBCO</u>	linpack	general-band
<u>CGBDI</u>	linpack	general-band
<u>CGBFA</u>	linpack	general-band
<u>CGBMV</u>	linpack	general-band
<u>CGBSL</u>	linpack	general-band
<u>CGECO</u>	linpack	general
<u>CGEDI</u>	linpack	general
<u>CGEEV</u>	d4	eigenvalues, eigenvectors
<u>CGEFA</u>	linpack	general
<u>CGEFS</u>	d2	linear-equations
<u>CGEIR</u>	d2	linear-equations
<u>CGEMM</u>	d1b	matrix-operations
<u>CGEMV</u>	d1b	matrix-operations
<u>CGERC</u>	d1b	matrix-operations
<u>CGERU</u>	d1b	matrix-operations
<u>CGESL</u>	linpack	general
<u>CGTSL</u>	linpack	general-tridiagonal
<u>CH</u>	eispack	
<u>CHBMV</u>	d1b	matrix-operations
<u>CHEMM</u>	d1b	matrix-operations
<u>CHEMV</u>	d1b	matrix-operations
<u>CHER</u>	d1b	matrix-operations
<u>CHER2</u>	d1b	matrix-operations
<u>CHER2K</u>	d1b	matrix-operations
<u>CHERK</u>	d1b	matrix-operations
<u>CHFDV</u>	e	interpolation
<u>CHFEV</u>	e	interpolation
<u>CHICO</u>	linpack	complex-hermitian
<u>CHIDI</u>	linpack	complex-hermitian
<u>CHIEV</u>	d4	eigenvalues, eigenvectors
<u>CHIFA</u>	linpack	complex-hermitian
<u>CHISL</u>	linpack	complex-hermitian
<u>CHKDER</u>	f	nonlinear-equations
<u>CHPCO</u>	linpack	complex-hermitian
<u>CHPDI</u>	linpack	complex-hermitian
<u>CHPFA</u>	linpack	complex-hermitian
<u>CHPMV</u>	d1b	matrix-operations
<u>CHPR</u>	d1b	matrix-operations
<u>CHPR2</u>	d1b	matrix-operations
<u>CHPSL</u>	linpack	complex-hermitian
<u>CHU</u>	c	elementary-functions, special-functions
<u>CINVIT</u>	eispack	
<u>CLBETA</u>	c	elementary-functions, special-functions
<u>CLNGAM</u>	c	elementary-functions, special-functions
<u>CLNREL</u>	c	elementary-functions, special-functions
<u>CLOG10</u>	c	elementary-functions, special-functions
<u>CMGNBN</u>	i2	partial-differential-equations
<u>CNBCO</u>	d2	linear-equations
<u>CNBDI</u>	d3	determinants
<u>CNBFA</u>	d2	linear-equations

<u>CNBFS</u>	d2	linear-equations
<u>CNBIR</u>	d2	linear-equations
<u>CNBSL</u>	d2	linear-equations
<u>COMBAK</u>	eispack	
<u>COMHES</u>	eispack	
<u>COMLR</u>	eispack	
<u>COMLR2</u>	eispack	
<u>COMOR</u>	eispack	
<u>COMOR2</u>	eispack	
<u>CORTB</u>	eispack	
<u>CORTH</u>	eispack	
<u>COSDG</u>	c	elementary-functions, special-functions
<u>COSQB</u>	j1	fast-fourier-transforms
<u>COSQF</u>	j1	fast-fourier-transforms
<u>COSQI</u>	j1	fast-fourier-transforms
<u>COST</u>	j1	fast-fourier-transforms
<u>COSTI</u>	j1	fast-fourier-transforms
<u>COT</u>	c	elementary-functions, special-functions
<u>CPBCO</u>	linpack	hermitian-positive-definite-band
<u>CPBDI</u>	linpack	hermitian-positive-definite-band
<u>CPBFA</u>	linpack	hermitian-positive-definite-band
<u>CPBSL</u>	linpack	hermitian-positive-definite-band
<u>CPOCO</u>	linpack	hermitian-positive-definite
<u>CPODI</u>	linpack	hermitian-positive-definite
<u>CPOFA</u>	linpack	hermitian-positive-definite
<u>CPOFS</u>	d2	linear-equations
<u>CPOIR</u>	d2	linear-equations
<u>CPOSL</u>	linpack	hermitian-positive-definite
<u>CPPCO</u>	linpack	hermitian-positive-definite
<u>CPPDI</u>	linpack	hermitian-positive-definite
<u>CPPFA</u>	linpack	hermitian-positive-definite
<u>CPPSL</u>	linpack	hermitian-positive-definite
<u>CPQR79</u>	f	nonlinear-equations
<u>CPSI</u>	c	elementary-functions, special-functions
<u>CPTSL</u>	linpack	positive-definite-tridiagonal
<u>CPZERO</u>	f	nonlinear-equations
<u>CQRDC</u>	d5	qr-decomposition
<u>CQRSL</u>	d5	qr-decomposition
<u>CROTG</u>	d1a	vector-operations
<u>CSCAL</u>	d1a	vector-operations
<u>CSEVL</u>	c	elementary-functions, special-functions
<u>CSICO</u>	linpack	symmetric
<u>CSIDI</u>	linpack	symmetric
<u>CSIFA</u>	linpack	symmetric
<u>CSINH</u>	c	elementary-functions, special-functions
<u>CSISL</u>	linpack	symmetric
<u>CSPCO</u>	linpack	symmetric
<u>CSPDI</u>	linpack	symmetric
<u>CSPFA</u>	linpack	symmetric
<u>CSPSL</u>	linpack	symmetric
<u>CSROT</u>	d1a	vector-operations
<u>CSSCAL</u>	d1a	vector-operations
<u>CSVDC</u>	d6	singular-value-decomposition
<u>CSWAP</u>	d1a	vector-operations
<u>CSYMM</u>	d1b	matrix-operations
<u>CSYR2K</u>	d1b	matrix-operations
<u>CSYRK</u>	d1b	matrix-operations
<u>CTAN</u>	c	elementary-functions, special-functions
<u>CTANH</u>	c	elementary-functions, special-functions
<u>CTBMV</u>	d1b	matrix-operations

<u>CTBSV</u>	d1b	matrix-operations
<u>CTPMV</u>	d1b	matrix-operations
<u>CTPSV</u>	d1b	matrix-operations
<u>CTRCO</u>	linpack	triangular
<u>CTRDI</u>	linpack	triangular
<u>CTRMM</u>	d1b	matrix-operations
<u>CTRMV</u>	d1b	matrix-operations
<u>CTRSL</u>	linpack	triangular
<u>CTRSM</u>	d1b	matrix-operations
<u>CTRSV</u>	d1b	matrix-operations
<u>CV</u>	l	statistics
<u>D1MACH</u>	r1	machine-constants
<u>D9PAK</u>	a	arithmetic-functions
<u>D9UPAK</u>	a	arithmetic-functions

Date and Revisions

Revision date -----	Keyword affected -----	Description of changes -----
18Mar96	<u>entire</u>	Text updated for SLATEC version 4.1. Adapted for LC (from NERSC).
31Oct91	background loading-slatec entire	New keyword for document comparisons. New loading instructions for UNICOS, CSOS. Text upgraded to cover SLATEC version 4.0.
30Nov87	entire	Text upgraded to cover SLATEC version 3.1. Page index added; keyword index expanded.
26Oct82	entire	First edition of new writeup.

TRG (18Mar96)

UCID-19631,19632,19633

Privacy and Legal Notice (URL: <http://www.llnl.gov/disclaimer.html>)

TRG (18Mar96) Contact on the OCF: lc-hotline@llnl.gov, on the SCF: lc-hotline@pop.llnl.gov